DOCUMENT RESUME

ED 029 673 52 LI 001 519

Cognitive Memory: A Computer Oriented Epistemological Approach to Information Storage and Retrieval. Interim Report, Phase I, 1 September 1967-28 February 1969.

Illinois Univ., Urbana. Coordinated Science Lab.
Spons Agency-Office of Education (DHEW), Washington, D.C. Bureau of Research.
Bureau No-BR-7-1213
Pub Date 30 Apr 69
Contract-OEC-1-7-071213-4557
Note-176p.
EDRS Price MF-$0.75 HC-$8.90
Descriptors-*Cognitive Processes, *Computational Linguistics, Computer Programs, Computers, *Information Processing, *Information Retrieval, Information Storage, *Man Machine Systems, Syntax

In contrast to conventional information storage and retrieval systems in which a body of knowledge is thought of as an indexed codex of documents to which access is obtained by an appropriately indexed query, this interdisciplinary study aims at an understanding of what is "knowledge" as distinct from a "data file," how this knowledge is acquired, and how this knowledge can be made effective through symbolic discourse between man and machine. The purpose is the development of cognitive memory systems which are capable of responding with structured information that matches the gap in the knowledge of the querist, rather than with the delivery of a "document," i.e., an accidental linguistic representation of the information about a particular fact, that may or may not cover the point in question. This report, arranged chronologically, describes the work done in each of the five report periods. The initial research action centered around the development of a linked data structure concept (called "cylinders," based on a use of "rings") and other technical aspects. Later, the study concentrated more on the fundamental interactions involved in cognition, attempting not to duplicate human intelligence but to design machines to accomplish results that are similar to the results of cognitive processes. A pilot information system-- "Rules of the Road" -- was developed for experimental use. (Author/RM)

ED029673

LI 001519

CSL COORDINATED SCIENCE LABORATORY

INTERIM REPORT
PHASE I

1 SEPTEMBER 1967 - 28 FEBRUARY 1969

ERIC/CLIS
JUN 30'69
UNIVERSITY OF MINNESOTA

COGNITIVE MEMORY

LI 001519

UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS

BR-7-1213
PA-52

OE-BR

INTERIM REPORT

PHASE I

1 September 1967 - 28 February 1969

COGNITIVE MEMORY

A Computer Oriented Epistemological Approach to Information Storage and Retrieval

USOE BUREAU OF RESEARCH
Grant No. OEC-1-7-071213-4557

PROJECT DIRECTORS:
Heinz Von Foerster
Robert T. Chien

Date of Issue
30 April 1969

COORDINATED SCIENCE LABORATORY

UNIVERSITY OF ILLINOIS

Urbana, Illinois

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

PREFACE

The following pages constitute the Interim Report of Phase I of the research activities sponsored under the auspices of USOE Bureau of Research Grant No. OEC-1-7-071213-4557, entitled Cognitive Memory; A Computer Oriented Edistemological Approach to Information Storage and Retrieval. This Interim Report is submitted in accordance with requirements specified under the above Grant, and covers the period from 1 September 1967 to 28 February 1969.

The first three Sections A (Abstract), B (Introduction) and C (Computers and the Language Problem) describe the activity as planned and as accomplished in general terms and in ascending order of details. We are particularly grateful for Mr. P. Weston who authored Section C. Since this interdisciplinary research program ranges over a great variety of topics, it was felt that individuals or individual groups working on a particular aspect of this problem should report their findings in their own words. Section D (Accomplishment Summary) is a collection of essays that reflect the activity of these individuals or groups. Although this collection will give a rather incomplete and spotty account of the progress of this project, it is hoped that the main lines of thoughts and achievements may emerge without too much opacity.

H.V.F.

R.T.C.

TABLE OF CONTENTS

		Page
	Preface	i
A.	Abstract	ii
B.	Introduction	iii
C.	Computers and the Language Problem	vii
D.	Accomplishment Summary	1-156

COGNITIVE MEMORY

A Computer Oriented Epistemological Approach to Information Storage and Retrieval

A. ABSTRACT

In contrast to conventional information storage and retrieval systems in which a body of knowledge is thought of as an indexed codex of documents to which access is obtained by an appropriately indexed query, this study aims at an understanding of what is "knowledge" as distinct from a "data file," how this knowledge is acquired and how this knowledge can be made effective through symbolic discourse between man and machine.

This approach is motivated by the recently rapid development of our insight into theory and neuropsychology of knowledge per se, i.e., theoretical and experimental epistemology (1)(2); into theory and neurophysiology of the processes that permit the acquisition of knowledge, i.e., cognitive processes (3)(4)(5); into theory and physiology of mechanisms that preserve this knowledge, i.e., "memory" as opposed to "record" (6)(7)(8)(9); and, finally, into the utilization of this knowledge through symbolic discourse in the form of natural language in a man-machine system in which each partner is entitled to make "queries", i.e., to pose problems to the other partner who may solve them by recourse to deductive or inductive reasoning (10)(11)(12)(13).

This approach is justified by an estimated order of magnitude improvement in the efficacy of information retrieval systems in as much as cognitive memory systems respond with structured information that matches the gap in a knowledge of the querist, rather than with the delivery of a "document," i.e., an accidental linguistic representation of the information about a particular fact, that may or may not cover the point in question.

B. INTRODUCTION

Current trends in the new science of information retrieval are to expand and to improve upon the traditional, non-mechanical, methods for organizing and searching information collections by applying present day computing technology. The searching of indexed document files and even to some extent the indexing work itself have been mechanized, while at the same time new and more useful indexing methods have been developed. Altogether, considerable research effort has already been expended, practical results have appeared, and research in this area is continuing.

Characteristic of virtually all this current retrieval research is the assumption that an information store must consist of a collection of documents, though various groups may choose to define the term "document" at different levels of text organization. Regardless of the way in which material is divided inside such a store, once the division has been made and an index established, the information content of the stored material has no further effect on the system. The documents become, as it were, so many numbered "black boxes," of which only the mind of a human reader can make any use. To gain access to a particular box amongst these boxes the user of such a system has to play a kind of guessing game. The system, unable to "understand" what the user wants to know, requires him to describe, in a (usually) very rudimentary form of language dictated by the document index, not what he wants to know but the kind of document which could have bearing on his problem. The description he generates constitutes two guesses, first that he knows how the system's indexers have interpreted the index language, and second that he has decribed a manageable set of documents that contain, among them, the information he is looking for. If he is wrong on either count he must guess again. Even after a good guess, the information must be sifted and collated from among a number of documents.

As reasonable as this position may seem, the necessity of "documents", i.e. detailed records of human linguistic output, is now for the first time--since the invention of writing--being thrown open to question by developments in the field of artificial intelligence. Computer programs designed in recent years by Lindsay (11), Raphael (12), and Bobrow (13), among others, have demonstrated that it is possible for a machine to admit information expressed in acceptable forms of natural language, store it in a reduced, non-linguistic form, and re-generate many acceptable linguistic outputs expressing not only the original input, but also numerous implications that can be inferred from the input by operating on the stored, non-linguistic representations of the input information with appropriate logical and syntactic rules and operators. At present these machines are rudimentary in many ways and do not cope at all with the important subtleties of natural language, but they do clearly point to a new foundation for retrieval research in which the information store contains, not "slots" into which inert records may be fitted, but a composite representation of a field of information from which, by internal rules and operations a system may generate a vastly larger variety of outputs than the number of documents which would be capable of expressing its original input. This form of information store we will call a "cognitive memory."

Should this form of system ultimately be realized in useful form it would have fundamental advantages over current systems. One of these refers to acceptable forms of requests. Presently, with document files, a "query" consists of an attempted description of a document set, using a crude form of language dictated by the indexing scheme and indexing terms, which is rather like going to a grocery store intending to buy milk, let us say, and asking for something that comes in bottles of a certain shape. Of course, the result may not be what was wanted, and the request is awkwardly indirect. In the normal use of language a question has the function of alerting someone to our desire for information and, most importantly, of describing the particular gap in our knowledge that needs filling. It is this latter, direct, form of query which is appropriate to a cognitive memory system. Similarly the output of a cognitive system can be made directly responsive to the requirement expressed in the query, and cannot, in fact take on the "canned" form of a document which is prepared remotely for its eventual users. A corollary of this has an important influence upon the direction of future work in that the system-user dialog is an essential feature of a cognitive information system.

There is a further important property inherent in cognitive memory storage which is independent of the foregoing man-machine interface features. This is that in it information is stored cumulatively and with common access on the basis of content. Provided that computer storage having both sufficient complexity and size can be developed, this will mean that facts which might have appeared in unlikely sources or as important but unstressed features of technical discussions could be released from their document "prisons" and made available to enrich the output information quality.

For these reasons in particular, and because of the broad applications of the results, we feel that pursuing the development of cognitive systems is presently the most important line of basic research which can be followed in the information sciences. The ground work has been laid, but it must be emphasized that the full realization of a useful cognitive memory system is still very far off due both to our limited knowledge of linguistic and cognitive structure as well as to present limitations of computing systems. Our goal is to attack these current problems and produce a "second generation" cognitive system design.

REFERENCES

1) D. M. MacKay: "Communication and Meaning--a Functional Approach" in Cross-Cultural Understanding: Epistemology in Anthropology, F.S.C. Northrop (ed.), Harper & Row, New York, pp. 162-179 (1964).

2) Warren S. McCulloch: "Postulational Foundations of Experimental Epistemology" in Cross-Cultural Understanding: Epistemology in Anthropology, F.S.C. Northrop (ed.), Harper & Row, New York, pp. 180-193 (1964).

3) J. Y. Lettvin; H.R. Maturana; W.S. McCulloch and W. Pitts: "What the Frog's Eye tells the Frog's Brain", Proc. I.R.E. 47, 1940-1951 (1959).

4) H. Von Foerster: "Circuitry of Clues to Platonic Ideation," in Artificial Intelligence, C.A. Muses (ed.), Plenum Press, New York, pp. 43-82 (1962).

5) H. Von Foerster: "Computation in Neural Nets," Currents Mod. Biol., 1, pp. 47-93 (1967).

6) J. J. Gibson: "The Problem of Temporal Order in Stimulation and Perception," J. Psychol., 62, pp. 141-149 (1966).

7) H. Von Foerster: "Memory without Record" in The Anatomy of Memory, D.P. Kimble (ed.), Science and Behavior Books, Palo Alto, pp. 388-433 (1965).

8) J.Z. Young: "The Organization of a Memory System", Proc. Royal Soc. 163, pp. 285-320 (1965).

9) J.Z. Young, personal communication (H.V.F.).

10) M. Minsky: "Steps toward Artificial Intelligence," Proc. IRE 49, 8-30 (1961).

11) R.K. Lindsay: "Inferential Memory as the Basis of Machines which Understand Natural Language," in Computers and Thought, E. Feigenbaum and J. Feldman (ed.), McGraw-Hill, New York, pp. 217-233 (1963).

12) R. Raphael: "A Computer Program which 'Understands'," Proc. AFIPS, F.J.C.C., 577-589 (1964).

13) D.G. Bobrow: Natural Language Input for a Computer Problem Solving System, Project MAC, Report No. TR-1, MIT, Cambridge, Mass. (1964).

C. COMPUTERS AND THE LANGUAGE PROBLEM

I. The Problem

With the current public faith in the unlimited power of technology, and with a firmly established but exalted image of computers as electronic "brains," there is already an established lore, presently of a humorous and fictional nature, connected with the "super computer," and apparently serious talk is now being heard regarding the problems men will face when mechanical rather than human intelligence rules the world and human labor becomes obsolete. Naturally, this optimism regarding the power of computers is shared by us who are trying to mechanize cognitive processes. But the curious truth is that the goal we seek with much energy and enthusiasm is already a household word while we have not yet fully clarified the basis for achieving it.

Let us begin to examine this unsettling situation by imagining what mechanized cognitive processes--the term "artificial intelligence" or "AI" for short, has been widely adopted for this idea--would look like from the outside, then moving to a discussion of what has been done, what significance it has, and what is left--if anything at all--to be done in the future.

Regarding the outward identifying characteristics of an artificial intelligence, the public idea is essentially the same as that of the researchers, and in substance amounts to the requirement that such a thing must act human. This was once stated by A.M. Turing in a form now widely known and accepted as the "Turing Test." This test postulates that a machine can converse with a human through any appropriate means, cathode ray displays, teletype, simulated voice, or whatever, and if the person, not knowing at the outset that a machine is at the other end of the channel, is still satisfied after however long he wishes to maintain the conversation that it could be a person at the other end, then the machine has passed the test and can be considered intelligent.

This criterion is sufficiently objective as originally stated to satisfy most demands of scientific method, and this is surely the reason for its continued acceptance for roughly thirty years. In practice, though, a loophole has shown up, and it is rather difficult to patch it up without losing the elegance and objectivity of the original. The trouble is that machines already exist which have passed the Turing Test on repeated occasions, but no one in the AI field, including their inventors, is willing to consider them very intelligent because they are rather simply organized, and, more importantly, all they can do is converse and only in a very restricted context, for instance in one type of psychiatric interview. On the whole the fact that these machines can lead their "patients" to believe they are talking with a human being, simply shows--objectively--that ordinary conversation often does not require much intelligence, and this was hardly the intent of the Turing Test. If the criterion were elaborated to include some stipulated range of topics, or various types of problem-solving behavior, etc., etc., the neatness of the original, and much of its objectivity would be lost.

The trouble is that intelligence cannot be pinned down by linking it to any one ability, a fact which was grasped by Binet some time before Turing's suggestion was made--though this is not meant to suggest that a standardized I.Q. test be adopted as a working goal. What is in fact the case is that AI cannot reasonably be expected to establish for itself a formal definition of intelligence because it is to a significant extent involved in the attempt to discover what human intelligence is, in the framework of information processing. This knowledge is needed to further the primary goal of designing machines to do what human brains do, though perhaps differently and perhaps eventually in better ways. It amounts to a difficult form of mimicry, and in common with all mimicry demands at least partial understanding of its object, sufficient at least to allow a translation from the domain of action of the original to that of the copy, which domains are in this case very different. We do not yet have the necessary understanding.

There is no point in comparing mechanical mimicry of intelligent behavior with "monkey see, monkey do," which should anyway be amended to read, "monkey see, monkey understand, monkey do." The difficulty comes right at the outset, with the monkey see stage, due to the great difficulty in actually observing the interior workings of intelligent behavior, rather than its external results. And nothing less than such an "interior view" will do. We are not as fortunate as a monkey copying a gesture, who comes supplied with a set of limbs and joints which can be set in roughly one-to-one correspondence with his human counterpart, and all of which can be directly seen in action. We, on the other hand, are given a computing machine to work with which has no obviously identifiable features in common with a human being, and therefore we have no model a priori for the workings we wish to copy.

The relevant behavior of the human being we are trying to model in the computer is observable only to the extent that we can provide him with inputs in the form of questions, problems, and other bits of language (if language is rather broadly defined), and he in turn can supply us with answers, solutions, and other relevant or irrelevant bits of language, which may include his own informal reports of the processes which led to his answers. While the latter may be of some direct benefit, it is clear that we are forced to work almost entirely from the outside and almost entirely through the medium of human language, which is of course itself a large part of the behavior we want to model. Simply in order to gets its "raw data" straight, AI has no choice but to attempt an anlysis of language, and in particular of the concept of meaning, because it is in this aspect, as we see it, that language reflects what we seek, the structure of intelligent behavior.

It is fundamental that a description of language by itself, no matter how objectively obtained or finely detailed it might be, could not suffice. The requirement is for a description of the relationship of language to cognitive processes. It is here that AI and current linguistic theory reach a point of divergence, because current linguistic theories of language structure appear to be predicated upon the exclusion of cognitive considerations, though this is a less rigid position now than in the past and the encroachment of cognitive categories into linguistic "deep structure" has been ever-present. Whether or not this divergence is real or long-lasting, it is a fact that AI is fundamentally involved with language and comes to the subject with its own point of view.

If we are to learn enough from this undertaking to be able to construct machines which are able to use human language then we must at least find out exactly what it is, at the cognitive level, which humans express in language, and then discover ways of representing and expressing such things in our machines. Neither task is easy.

II. What has been done

In groping for solutions to these problems, computers have been indispensable aids and, occasionally, companions.

Of course the familiar arguments in favor of computers hold here, i.e., that, in reducing a system to a computer realization, considerable crystallization of concepts is demanded (partially offset by obfuscation introduced by trivial system and program details) and fatal flaws come relatively quickly to light once the machine is turned loose. Moreover at this moment there seems to be no practical alternative to relying on computers to test the complicated and largely non-mathematical hypotheses that AI deals with. The history of AI could be written in terms of a fairly small number of simulation programs, a good portion of which were designed to deal with some aspect of the problem of meaning.

The present and past programming efforts fall into three somewhat overlapping categories:

a) structured data base fact retrieval systems,

b) deductive question-answering systems,

c) associative memory systems.

Categories (a) and (b) have only recently begun to split off from a common root and differ at present mainly on matters of emphasis. The central idea of both is to establish in computer storage a formal data base consisting of a catalog of items along with a system of categories upon them and statements of the relationships between them, and to use this store of information as the domain of discourse in a linguistic interaction between a computer program and a human being. The difference between the two approaches (a) and (b) lies in the fact that whether fact retrieval or deductive question-answering is the more important goal, the language used in the interaction tends respectively to be a "subset" of natural language, or a formalized language. Put perhaps more realistically, a tendency has developed to concentrate separately on the two tasks of increasing the language capability of such a program and on increasing the depth of its deductive power.

In category (c) there is more emphasis on lexical structure and the wide range of associations characteristic of human concepts, and less on purely deductive methods. This type of approach appears more directly germaine to the development of closer approaches to natural language for man-machine interaction, though all of the above and more will eventually be needed.

Rather than reciting details of the programs, which are available in the literature, I will simply offer some opinions on how I believe they relate to the AI picture. First, one principle is amply borne out by the existing work, and that is that when human functions are simulated by machines they will be realized through forms of organization very different from the human ones. For example, the programs which use restricted forms of natural language behave nothing like inexperienced human speakers, but would be better described in human terms as extreme examples of the "one-track" mind, in the sense that only one context exists in the "world" of such a program and that is the one in which all statements are well-formed sentences which convey either simple facts or questions, and which bear upon the given data base. In contrast, and it is not meant to be a derogatory contrast in either direction, a human child who has not yet mastered all of the possibilities of his language is still likely to know all about such things as truth, falsity, fiction, lies, wishes, guesses, promises, emphasis, emotional overtones, fragmentary utterances, and so on, and in fact he is unable to interpret language without dealing with such contextual matters.

This contrast suggests an analogy with machines for performing physical functions. The potential benefits of power and speed which are available mechanically are generally only realized by concentrating on limited functions and using the means which are most appropriate to the given end. The result is usually a device which far exceeds the power of direct human labor, and similarly the promise of adapting computers to specialized cognitive functions and achieving better-than-human performance is now very real.

In a different vein, it is legitimate to ask, in light of the limited use of language which is involved in the machines mentioned so far, how and to what extent they actually deal with meaning. Such a question could not be finally answered until we were all sure that we understood the nature of linguistic meaning, but short of that, an answer can be offered in the form of a description of the behavior which the programs copy, and of the means they use to do so.

One aspect of intelligent understanding of language appears to be primarily involved, and this is that in the context of factual information from a reliable source a human can demonstrate his understanding of a body of presented data by answering questions, regarding the explicit facts as given and also what can be logically deduced from them. While this does not exhaust the list of possible human responses to language, it is surely a basic one and sufficiently difficult to mimic adequately that it has formed an interesting and productive research area for simulation studies.

Perhaps the most salient feature of these AI programs is their approach to the representation of meaning, which approach, the limited scope of current research notwithstanding, does in its most general aspect typify the unique assumptions which AI brings to the language problem. One of these is that meaning is not taken to reside in an ultimate representational level of language, but to be based in the (assumed) more general mechanism whereby all of the organism's knowledge of external and internal reality is represented, whether this knowledge is derived from linguistic or perceptual channels. This mechanism must include the dynamic processes which are responsible for various forms of inference, logical and intuitive, and for the active building of models in the course of experience. The AI notion of the interpretation of a piece of text, for example, does not stop at the disambiguation of syntactic and semantic forms but further entails the construction of a model for the content of the text which will serve to connect it with the remainder of the understanding organism's knowledge and to mediate as wide a variety of useful inferences upon that knowledge as possible.

On the basis of this approach to interpretation, it is natural to expect that the model-building and inference processes should themselves play an integral part in facilitating the interpretation of natural language utterances and this viewpoint is indeed a common one in the AI field, and is shared by myself. This also is the basis of a further philosophical difference between AI and linguistic theory, for from the purely linguistical point of view Occam's razor appears reasonably to demand that modeling and inference processes be viewed as excess baggage, while from the AI viewpoint language is an empty concept if it is not related to its function in the overall picture of intelligent behavior.

Aside from occasional misunderstandings, the most important negative result of this divergence has been the limited applicability of advances in linguistic theory to progress in AI, though it is true that the exercise of ingenuity has resulted in the effective adaptation of linguistical notions of semantics in limited scale simulation projects (see Simmons *et al*., 1968). In the long run, however, it seems reasonable to expect that AI must develop its own ways of dealing with syntactic and semantic problems, and that these may,eventually, contribute in their turn to linguistic theory.

These speculations aside, AI programs already have succeeded to some degree in coupling language to other aspects of intelligence; some of the principal accomplishments being retrieval of facts from a structured data base in response to questions stated in a fairly free subset of English, answering questions requiring moderately difficult deductions from a set of given facts, and the building of data base from restricted natural language statements. Here one may include the important step toward adaptability of allowing new categories and relations to be introduced through definitions supplied in the same interaction language used for queries and facts. This much is substantive accomplishment and is beginning to pass from the state of laboratory curiosity to that of working tool. But there is still some distance to go before there will be programs with even child-like facility in the everyday use of natural language.

III. Present Activity

The following pages, entitled Accomplishment Summary, give a historical account of the work performed under the auspices of this grant. It was felt that the activity of each participant should be described in his own words. It is hoped that the collection of essays that follow present a coherent line of thoughts that aim toward a unified concept of cognitive processes and their possible implementation in computer soft and hard ware.

References

Bobrow, D.G., "A Question-Answering System for High School Algebra Word Problems," AFIPS Conf. Proc.,26, Fall Joint Computer Conference, 591-614 (1964).

Chomsky, N., Aspects of the Theory of Syntax, MIT Press, Cambridge (165).

Fillmore, C.J., "Toward a Modern Theory of Case," in Report No. 13, Project on Linguistic Analysis, Ohio State University (1966).

Green, C.C. and B. Raphael, "Research on Intelligent Question-Answering System," Scientific Report No. 1, Stanford Research Institute (1967).

Katz, J.J. and A. Fodor, "The Stucture of a Semantic Theory," in The Structure of Language, J.J. Katz and A. Fodor (eds.), Prentice-Hall, N.Y., pp. 479-518 (1964).

Kellog, C.H., "CONVERSE - A System for the On-Line Description and Retrieval of Structured Data Using Natural Language," Document SP-263s, System Development Corp., Santa Monica, California (1967).

Lindsay, R.K., "Inferential Memory as the Basis of Machines which Understand Natural Language," in Computers and Thought, E.A. Feigenbaum and J. Feldman (eds.), McGraw-Hill, N.Y., pp. 217-233 (1963).

Quillian, R., Semantic Memory, Ph.D. Thesis, Carnegie Institute of Technology (1966).

Raphael, B., "A Computer Program Which 'Understands'," Proc. AFIPS, Fall Joint Computer Conference, 577-589 (1964).

Weizenbaum, J., "ELIZA - A Computer Program for the Study of Natural Language Communication Between Man and Machine," Comm. of the ACM 9, 1, pp. 36-44 (1966).

Woods, W.A., "Procedural Semantics for a Question-Answering Machine," in Proc. AFIPS, Fall Joint Computer Conference, 457-471 (1968).

D. ACCOMPLISHMENT SUMMARY

D. ACCOMPLISHMENT SUMMARY

TABLE OF CONTENTS

	Chapter Pagination	Running Pagination
I. Accomplishments from 9/1/67 - 11/30/67		1
II. Accomplishments from 12/1/67 - 3/31/68		4
A. Preface	1	5
B. Major Activities and Accomplishments During Report Period	3	7
1. Organizational	3	7
(i) Weekly Workshop	3	7
(ii) IGLIS Document System	4	8
2. Technical	7	11
(i) Language Acquisition	7	11
(ii) An Investigation of the Normalization of Natural English Text	8	12
(iii) Work Frequency Study	11	15
(iv) Theorem Proving Techniques	12	16
(v) Time-Space Scheduling	15	19
(vi) Textual Macrostructure	16	20
(vii) Studies of Access Rates to Densely Interconnected List Organized Data	18	22
III. Accomplishments from 4/1/68 - 7/31/68		23
A. Preface	1	24
B. Major Activities and Accomplishments During Report Period	3	26
1. "Cylinders," A Linguistic Data Structure Based On a Novel Use of "Rings"	3	26
(i) Introduction	3	26
(ii) Ring Structures	7	30
(iii) Cylinders	11	34
(iv) Examples of Use	19	42
(v) Summary	20	43
2. An Objective Function for the Scheduling Routines of a Time-Sharing System	23	46
3. Cognitive Memory	31	54
4. A Pilot Information System: "Rules of The Road"	31	54
(i) Introduction	31	54
(ii) Preprocessing	32	55
(iii) Steps Towards a Relational Structure	35	58

(iv) On the Recognition of Sentence Types...36 59
(v) Syntactic Processing....................38 61
5. Textual Macrostructure......................40 63

IV. Accomplishments from 8/1/68 - 11/30/68 65

A. Preface... 1 66

B. Major Activities and Accomplishments During Report Period.................................. 3 68
1. A Pilot Information System R2: "Rules of the Road"................................ 3 68
(i) Introduction............................. 3 68
(ii) Query Classification..................... 3 68
(iii) Syntactic Analysis......................10 75
(iv) Context Modeling........................20 85
(v) Concept Processing......................24 90
2. Grammars and Relational Structure...........27 93
3. Investigation of Fundamentals of Nonlinguistic Cognition...........................31 97
4. Cognition and Heuristics....................34 100
5. Machine Architecture for Information Retrieval.................................36 102
(i) Introduction............................36 102
(ii) Investigation of the Processor of Lee and Paull..........................38 104
(iii) A New Associative Memory Processor.....44 110
6. Studies of the Mathematical Theory of Cognition..................................49 116
(i) On the Forms of Equations Associated With Inductive Inference Computers.....49 116
(ii) On a Class of Nonlinear Property Filters..............................53 120

V. Accomplishments from 12/1/68 - 2/28/69

A. Preface... 1 131

B. Major Activities and Accomplishments During Report Period.................................. 3 133
1. Research on the "R2" System.................. 3 133
(i) Question Analysis Techniques........... 3 133
(ii) Concept Processing..................... 6 136
(iii) Context Modeling....................... 8 138
(iv) Syntactic Processing................... 9 139
2. Semantic Compiler...........................15 145
3. Basic Concepts in Cognition.................18 148
(i) Review.................................18 148
(ii) Present Work...........................18 148

4. Associative Processor........................20 150
(i) Introduction.............................20 150
(ii) Previous Processors......................21 151
(iii) Present Results..........................23 153

I. ACCOMPLISHMENTS FROM 9/1/67 - 11/30/67

Item 1.

MAJOR ACTIVITIES AND ACCOMPLISHMENTS DURING REPORT PERIOD

At the beginning of the present period, the research group undertook the initial definition of organizational structures and functions for the senior staff, the research assistants, and the general membership of IGLIS (Interdepartmental Group on Library and Information Sciences). Subsequently, two principal foci of activity have centered respectively around weekly meetings of the senior staff and weekly joint meetings of the senior staff and research assistants. At the outset, the latter meetings have necessarily assumed primarily the format of didactic workshops as sucessive major aspects of the research program have been presented. The need to be met in this respect is to establish a thorough understanding of the principles on which the program is based, as these principles tend to be fundamentally different from those of most current information retrieval research with which the assistants are already familiar.

While the work and meetings of the senior staff have also reflected a need in part to review the common general background and to facilitate the articulation of the different approaches of individual members, their interaction has already been channeled into several areas of the more substantive issues to be addressed. To date, the latter have specifically included the following technical aspects: (1) Operational definitions of cognitive memory system components, (2) Encoding graphical lines into symbol chains, (3) Classification concepts in cognitive processes, (4) Non-linguistic cognitive representation of events mappable into verbal descriptions (as well as events themselves), (5) Ring and cylinder representation in computer memory of complex relational structures, and (6) Numerical estimates of computer search speeds with data involved in cognition experiments.

In illustration of the next level of detail on which each of these topics might be described, the summary abstract prepared for the last named effort may be cited as an example:

> "During this report period, a simulation of a computer system searching highly structured data was performed. The simulated system is a near-conventional central processor plus a magnetic disc store plus a special (practical) scheduling mechanism. The purpose of this simulation is to provide numerical estimates of the search speed of immediately practical computer systems when searching data of the type involved in cognition experiments.
>
> The results show that the scheduling mechanism can provide search rate improvements of 4 to 6 times the rate with no scheduler.
>
> These results per se are not viewed as of especial importance, but they do provide values against which other system designs can be compared. The simulation of other systems is proceeding." (S. Ray and B. Wang)

II. ACCOMPLISHMENTS FROM 12/1/67 - 3/31/68

Preface

The following pages give a brief account of the activity associated with the study on cognitive memory during the second report period from 1 December to 29 February 1968. Under Paragraph A in Item (i) some features of the inter-disciplinary organization of this project are described, while Paragraph B gives technical details.

The outstanding contributions to this study by Mr. Jerrold Sadock and Professor Arnold M. Zwicky from the Department of Linguistics, who tirelessly and patiently addressed themselves to this group in numerous discussions and lectures, is herewith acknowledged with great appreciation and gratitude.

H. Von Foerster

ACCOMPLISHMENTS FROM 12/1/67 - 3/31/68

Table of Contents

Page

Preface..1

Major Activities and Accomplishments During Report Period.......3

1. Organizational..3

   (i) Weekly Workshop.....................................3
   (ii) IGLIS Document System..............................4

2. Technical...7

   (i) Language Acquisition................................7
   (ii) An Investigation of the Normalization of Natural English Text.....8
   (iii) Work Frequency Study..............................11
   (iv) Theorem Proving Techniques.........................12
   (v) Time-Space Scheduling...............................15
   (vi) Texual Macrostructure..............................16
   (vii) Studies of Access Rates to Densely Interconnected List Organized Data.....18

MAJOR ACTIVITIES AND ACCOMPLISHMENTS DURING REPORT PERIOD

1. ORGANIZATIONAL

(i) Weekly Workshop - F. P. Preparata

In order to develop a common background and to familiarize the members of the group with the state-of-the-art of cognitive processes, a weekly workshop has been established. The meeting usually consists of a semiformal presentation--offered by a group member or, occasionally, by an invited speaker--as a review of current research in the area. The presentation is then followed by a discussion aimed at highlighting outstanding features, promises, and limitations of the work being examined.

The past workshops have dealt mainly with the representation of relations (and connected hardware and software general problems), the theory of transformational grammars, and the representation of semantic information. Although the workshop has not yet evolved into a "creative" phase, nonetheless its function in sensitizing the group members to the various problems of the cognitive memory has been highly successful.

(ii) IGLIS Document System - D. E. Carroll

During the present report period, one line of organizational activity has involved the establishment of a document distribution and bibliographic searching system to meet the needs of individual members of the IGLIS (Interdepartmental Group on Library and Information Sciences) personnel engaged on the Cognitive Memory Project. Copies of the forms used for document routing and service requests are attached. To date some 25 items have been processed on a trial basis, and the initial responds is to suggest the utility of the system and the desirability of expanding the depth and scope of the operation in the future.

From: (Name) ______________________ Item No. ______________

(Office) ______________________ Date ______________

To: D. E. Carroll
IGLIS Information Office
Room 3-107 Coordinated Science Lab

Subj: Action request concerning routed item

Instructions:

Indicated below any desired action you wish taken concerning the item routed. For immediate response, return this form directly to the Information Office (or phone 333-6448 or 333-0646).

Desired Action:

_____ Please xerox a duplicate for my use.

_____ Please xerox page(s) _____ for my use.

_____ Please compile and route a bibliography of other works by the author(s) of this item.

_____ Please compile and route a bibliography on the topic referenced on page(s) ____ cited as ______________

_____ Please procure and route a copy of the document referenced on page(s) ____ by/or titled ______________

_____ Please route the enclosed item also to the following addresses (or class(es) of addressees):

_____ Other desired actions. (Please specify).

Form B1--10 Jan 68

FROM: IGLIS Information Office
Room 3-107
Coordinated Sciences Laboratory

Item No.______________

Date______________

TO: IGLIS Members (as addressed below)

SUBJECT: Document Routing for Cognitive Memory Project

Instructions:

a. After examination, please cancel your name and forward the enclosed item within 2.5 days to another of the checked addressees below. (Route first preferably to any remaining addressee in your own building.)

b. If you cannot attend to the enclosed item within 2.5 days, please forward without cancelling your name (to ensure re-routing for your later attention). Re-routing will continue until all checked addressees are cancelled.

c. Indicate any desired action concerning the item on one of the accompanying action-request forms. The latter may be returned separately (if so desired for immediate response) to Room 3-107 CSL (or phone either 333-6448 or 333-0646).

Addressees:

IGLIS I

_____ H. Von Foerster....(216 Elec. Eng. Res. Lab.)
_____ R. Chien...........(3-101 Coord. Sci. Lab.)
_____ S. Ray.............(277 Dig. Comp. Lab.)
_____ B. Carss...........(805 W. Pennsylvania)
_____ F. Preparata.......(3-103 Coord. Sci. Lab.)
_____ J. Sadock..........(1116 W. Illinois)
_____ P. Weston..........(3-119 Coord. Sci. Lab.)
_____ A. Zwicky..........(309C Davenport Hall)

IGLIS II

_____ H. Bielowski.......(3-119 Coord. Sci. Lab.)
_____ K. Biss............(3-110 Coord. Sci. Lab.)
_____ P. Duran...........(3-121 Coord. Sci. Lab.)
_____ J. Harris..........(A15 Library)
_____ C. Hartman.........(3-110 Coord. Sci. Lab.)
_____ C. Kelley..........(3-113 Coord. Sci. Lab.)
_____ F. Ryan............(128 Library)
_____ F. Stahl...........(3-113 Coord. Sci. Lab.)
_____ B. Wang............(3-120 Coord. Sci. Lab.)

Addressees for further disposition on completion of above routing:

D. Carroll.........(3-107 Coord. Sci. Lab.)

2. TECHNICAL

(i) Language Acquisition - P. Weston

In September, the long range research goal was adopted of demonstrating the acquisition of some simplified form of natural language by a computer program. A principle requirement for achieving this, in our analysis of the problem, is the use of direct man-machine interaction, and furthermore a system at the interface which is at once simple enough for the machine to be capable of dealing with the structures, relationships, and operations involved and yet rich enough to be of some interest to the human participant.

The initial research steps have been directed toward programming-tool development in two simultaneous and at this stage mutually beneficial lines. Most importantly, a linked data structure concept, which we have called the "cylinder," has been conceived and implemented. This principle of data structuring allows multiple circular modes of search within a single structure, uses only a single principle of construction at all points in a complex structure, and needs what seems to be a near minimum in terms of memory space reserved for bookkeeping and other overhead uses.

In a developmental version, cylinders were used in the program XBAR (mentioned below) in the period September through November. In the period following, the experience gained with the first version and the new availability of the CSL7 list processing language on the 1604 have allowed the development of a generalized and improved form of the cylinder system which we will now be able to employ extensively in the next stage of programming work.

To aid in the development of the cylinder structure concept in its first form and simultaneously to lay a planning basis for an adequate interface system, a modest computer display program of the Sketch-Pad species was constructed, using cylinders to form the requisite data structure. This program, called XBAR, was brought to a level of usefulness which fulfilled its developmental function regarding both the data structures and their relationship to the interface design. With this accomplished, and with the availability of the CSL7 list processing language, the way is now clear to complete the interface work and proceed with preliminary development on the next phase of the project which is realization of the information processing programs to accept and analyze information coming in both graphic and linguistic form from the interface.

(ii) An Investigation of the Normalization of Natural English Text -

R. T. Chien, K. Biss

The purpose of this work is to investigate methods of converting natural English text into a canonical form, that when stored in a computer, the machine will be able to answer questions based on the normalized text in storage.

Following Garvin[1], we divide the set of sentences of a document into two types; predications and non-predications.

---

[1] Paul L. Garvin, "Research in Semantic Structure," Thompson Ramo Woolridge, Inc., R. W. Division, Technical Note No. 6, Jan. 15, 1963.

Predications are sentences which have the form "A is in some way a function of B," and non-predications are sentences which do not have this structure. For example the sentence "The SIR model is the collection of data which SIR subprograms may refer to in the course of question-answering." is a predication where the function "equality" is expressed by the word "is," but the sentence"'understanding' is difficult to define" is not a predication. In this work we deal strictly with predications.

A particular function in a predication can be expressed by many different words. In order to normalize the text we will replace words that express a function by the function itself. Thus in the sentence, "A horse is an animal," the function equality, which is expressed by the word "is," is substituted into the sentence. The sentence thus becomes "A horse equals an animal." The word "is" is called the original form and the word "equals" is the replacement.

Predications may appear in the text in several forms, i.e. aRb, abR, $bR^{-1}a$, Rab, etc. where a and b are "objects" and R is the relation between a and b. Predications of the form of aRb are left alone but all other forms are paraphrased so that they will be of the form aRb. For example, the sentence "Backus notation represents the grammar of a language as a set of definitions of grammatical structures," is changed to read "In Backus notation a set of definitions of grammatical structures represents the grammar of the language." where R is "represents" in this case.

In an effort to gain further insight into the usefulness of predications we surveyed three articles. These articles were: 1) Academic Disciplines - The Scholarly Dispute over the Meaning of Linguistics - *Time*, February 16, 1968; 2) Description of Syntax-directed Translator--C. M. Reeves - *Computer Journal*; 3) A Computer Program Which "Understands" - Bertran Raphael - AFIPS Conference, 1964.

Analysis of these articles produced several noteworthy results. First, non-technical text such as found in *Time* magazine contained very few predications, and the predications that did exist were hard to recognize. Second, in technical articles, excluding the abstract and introduction, most sentences were predications (about 70%) which were fairly easy to recognize. These predications were mostly of the form aRb and only a few had to be paraphrased. For example, a paragraph from B. Raphael's paper reads, "The SIR model *is structured by* means of property-lists (sometimes called description lists). A property-list *is* a sequence of pairs of elements, and the entire list *is associated* with a particular object. The first element of each pair *is* an attribute applicable to a class of objects, and the second element of the pair *is* the value of that attribute for the object described," where the function words are underlined. This paragraph was then changed to read, "The SIR model *is structured* by means of property lists (sometimes called description-lists). A property list *equals* a sequence of pairs of elements. The entire list *is associated with* a particular object. The first element of each pair *equals* an attribute applicable to a class of objects. The second element of the pair equals the value of the object described." where the replacement forms are underlined. Third, in the technical articles the non-predications did not carry very much information and in fact, some of the non-predications could be dropped from the text with almost no loss of information. For example in the paragraph "Most readers will be aware that the official document which defines the ALGOL language (Backus et. al., 1963) makes use of a special notation, the so-called Backus normal form, for defining the grammar, or rules of syntax, of ALGOL. This meta language is simple, powerful, and general. It is the basis for most formal specifications of current programming languages. We shall take it as our starting point.",the non-predication "We shall take it as our starting point" could be dropped from the text and never be missed.

In the future we intend to write a program to answer questions based on this idea of normalized sentences in a text.

(iii) Work Frequency Study - B. W. Carss

During the past six months, the discussions of the Cognitive Memory group has shown how relatively little is known about modern English language usage. It therefore, appears to be profitable to tackle the problem from two aspects.

a) To carry out a study of word frequency and word combination frequency in contemporary English. The most recent study of word frequency, etc. was carried out in 1931, and the book reprinted in 1944. A series of computer programs have been written to perform the necessary analyses automatically. A sample of approximately 100,000 has been assembled and processing will commence shortly.

b) To establish a data bank of spoken language. This data bank will make it relatively easy to compare written language with spoken language. Hopefully, we can begin to look for correlations between written language, spoken language and semantics.

The work proposed under (b) will be funded by another source and is mentioned here as being directly related to the cognitive memory project.

(iv) Theorem Proving Techniques - R. T. Chien, C. Hartman

Application of predictive calculus, in particular the theorem proving techniques, to improve the deductibility power of the computer have been studied. Based on the results of many projects in recent years, the Robinson resolution principle[1] in conjunction with the Herbrand approach[2] to theorem proving seems most effective because of its generality and its relative simplicity as it consists of just one simple inference principle. Its main advantage lies in its ability to avoid one of the major combinatorial obstacles to efficiency which have plagued other theorem-proving procedures. The basic approach of Herbrand proof procedures is to try to construct a model that satisfies the negation of the statement (more generally "well-formed formulas" (wff's)) to be proved. First the negation of the wff is put into a standard form. Then the resolution principle is applied to deduce some other wff, such that the original wff is satisfiable only if its descendants are all satisfiable. In its proportional form the Robinson resolution principle is illustrated as follows: $(P\vee B)\wedge(-P\vee C)\wedge D$ is inconsistent if and only if $(B\vee C)\wedge D$ is inconsistent. Where P is a proposition (in general and Atomic Formula); B,C,D are wff's and -P is the negation of P; $\vee$ stands for "or" and $\wedge$ stands for "and." In other words we have eliminated P. Let us give an example in the propositional calculus and apply the rule of inference. We know that: "John is a friend of Peter or John is a friend of Paul or friend of both" is a true statement and that "John is not a friend of Peter" is a true statement also. Let P, B stand respectively for "John is a friend of Peter", and "John is a friend of Paul." In a statement form we have that $(P\vee B)\wedge -P$ is true and by application of the rule of inference we can conclude that B is true. So we know that "John is a friend of Paul" is a true statement. In other words we can say that the truth table of $(P\vee B)\wedge -P$ depends only upon the truth values of B.

---

[1] J. A. Robinson, "A Machine-Oriented Logic Based on the Resolution Principle," JACM, Vol. 12, No. 1, January 1965.

[2] E. Mendelson, Introduction to Mathematical Logic, (von Nostrand, 1964).

A more realistic example would be to show that in a group a left inverse is also a right inverse. The axioms of a group that we are going to need for the proof are:

1. e.x=e (existence of left unity)
2. I(x).x=e (existence of left inverse)

3. $(x.y).z=w \rightarrow x.(y.z)=w$ (associative law)

4. $x.(y.z)=w \rightarrow (x.y).z=w$ (associative law)

For convenience the associative law was split in two. We want to prove that $x.I(x)=e$ (conclusion) is a true statement. Let the predicate of three arguments $P(x,y,z)$ be interpreted as $x.y=z$. In the disjunctive form the theorem is written in the following way:

1. $P(e,x,x,)$
2. $P(I(x),x,e)$
3. $-P(x,y,u) \vee -P(u,z,w) \vee -P(y,z,v) \vee P(x,v,w)$
4. $-P(y,z,v) \vee -P(x,v,w) \vee -P(x,y,u) \vee P(u,z,w)$

and the conclusion to be proved will take the form $P(x,I(x),e)$. In this particular case the single existential quantifier has no dependence on the universal quantifiers, hence leads to the constant function "s" when this existential quantifier is replaced by a function symbol. The theorem is proved in the following steps:

1. $P(I(I(s)),I(s),e)$ by axiom 2
2. $P(e,s,s)$ by axiom 1
3. $P(I(s),s,e)$ by axiom 2
4. $-P(I(I(s)),I(s),e) \vee -P(e,s,s) \vee -P(I(s),s,e) \vee P(I(I(s)),e,s)$
   by axiom 3, taking $(I(I(s)),I(s),s)$ for (x,y,z)
5. $P(e,I(s),I(s))$ by axiom 1
6. $-P(e,I(s),I(s)) \vee -P(I(I(s)),I(s),e) \vee -P(I(I(s),e,s) \vee P(s,I(s),e)$
   by axiom 4, taking $(I(I(s)),e,I(s))$ for (x,y,z)
7. $-P(s,I(s),e)$ by the negation of the conclusion
8. Since 1,2,3,4 are always true so $1 \wedge 2 \wedge 3 \wedge 4$ is true;

applying the resolution principle we get that P(I(s),e,s) is true.

9. Since 1,5,6,8 are true so is $1\wedge5\wedge6\wedge8$ and by the resolution principle we get that P(s,I(s)e) is true.

10. From 7 and 9 we have $P(s,I(s),e)\wedge-P(s,I(s),e)$ is true, what is inconsistent and the negation of the conclusion is false so the conclusion is true. In this particular case we really get the conclusion in step 9 but in general we get $A\wedge-A$ is true where A is any axiom or true clause, that is we deduce the empty formula.

In the application of this procedure to our particular problems, we feel that we can make significant improvements, especially in the relevancy criteria, that is, what should be the most appropriate axiom or theorem that should be used in the next step of the proof procedure. It seems that until now no powerful method has been developed in this sense. For example, an axiom or theorem to be used in the next step should take into account its generality, its complexity and its usefulness based on the previous experience of the system.

(v) Time-Space Scheduling - H. Bielowski

The problem of handling relational data structures with secondary sequential files suggested the study of the problem of time/space scheduling the activities in a real time computer system.

A general performance criterion for computers in form of a revenue-objective has been found. Revenue/response time-functions have been defined as a characteristic feature of computational tasks. Departing from there, four operational subgoals for the scheduling routines of a generalized computer system have been stated.

The implications of these subgoals have been critically related to recent publications. In the light of the stated objective, the state of the art of Main- and Input/Output scheduling appeared to be unsatisfactory.

Directions in which further work might find success have been found. The suggestions pivot around the statement of relevant items of information on the in-waiting and in-progress computing tasks as well as on the state of the system, information, which should be used by the scheduling programmers in order to enable them to effective scheduling decisions. The matter has been discussed in a 22-page paper.

(vi) Textual Macrostructure - D. E. Carroll

This line of effort has been addressed to the formulation and initial exploration of a theoretical problem relating to the "macrostructure" of at least some components and aspect of a cognitive memory system when operating on documentary textual information. The conjecture is that the natural language representation and communication of textual information presumes some hidden or underlying structures and relational or classificatory processes (heretofore undescribed) which would appear essential to at least certain aspects of the cognition and synthesis of information. It is believed that these structures and processes (however defined and acquired, perhaps by cultural convention) must be present to accomplish such tasks as the synthetic reduction of two or more related through differing accounts of a topic to a single unified account characterized by logical clarity and minimum redundancy.

It would appear that such reductive transformations must operate on the natural sentence and phrase levels, that a test of semantic equivalence and semantic relation (or distance) is required, and that a set of programmatic conventions must obtain with regard to relational or classificatory processes. Assuming for the present some effective means of testing for semantic equivalence and relation, work is presently being focused upon an inductive approach which will hopefully yield a useful corpus of inferences concerning the postulated programmatic conventions together with at least some tentative rules concerning their typology and applications.

(vii) Studies of Access Rates to Densely-Interconnected List-Organized Data - S. Ray & B. Wang

The preliminary investigation of the rates at which cognitive-type data (densely interconnected lists) can be accessed with a near-standard computer system was completed. The system was configured as a 3rd generation computer with the data largely contained on a magnetic disc. When searching through 10-20 lists in an order which minimized access time, the results showed that only some 100 data block per second could be accessed. This would correspond to (perhaps) 300-500 lines per second, the uncertainty being due to the lack of hard data concerning the number of lines per data block.

A second phase of the investigation was concerned with manual estimates of search rates in the case of the ILLIAC IV, where there are 256 arithmetic units with 4 megabytes of random access memory. In a random-access memory with coordinate-address links, list data can, of course, be processed at approximately the inverse of the memory access time (1 to 2 million lines per second in this particular case). This is completely uninteresting, however, since any standard computer with the same size core memory could perform equally well.

More interesting is the case in which the 4 megabyte memory is treated as a content-addressable structure. Here, the parallelism of the machine can be utilized. Under these conditions, about 1000 links per second could be processed. It remains to be determined whether the elegance of the content-addressable approach can be economically justified as against the faster but less manageable system of coordinate links.

III. ACCOMPLISHMENTS FROM 4/1/68 - 7/31/68

Preface

The following pages give a brief account of the activity associated with the study on cognitive memory during the third report period from 1 March to 31 May of 1968. Not reported in this account is the preparation of various scientific papers that represent the first tangible harvest of this basic investigation. Upon completion, these will be submitted under separate cover as Special Technical Reports.

Again, the contribution of members of the Department of Linguistics, of Mathematics, of Anthropology and of Psychology, whose association with this project is by interest and enthusiasm, rather than by contract, is herewith acknowledged with great gratitude and appreciation.

H. Von Foerster

ACCOMPLISHMENTS FROM 4/1/68 - 7/31/68

Table of Contents

Page

Preface..........1

Major Activities and Accomplishments During Report Period..........3

1. "Cylinders," A Linguistic Data Structure Based on a Novel Use of "Rings"..........3
    - (i) Introduction..........3
    - (ii) Ring Structures..........7
    - (iii) Cylinders..........11
    - (iv) Examples of Use..........19
    - (v) Summary..........20
2. An Objective Function for the Scheduling Routines of a Time-Sharing System..........23
3. Cognitive Memory..........31
4. A Pilot Information System: "Rules of the Road"..........31
    - (i) Introduction..........31
    - (ii) Preprocessing..........32
    - (iii) Steps Towards a Relational Structure..........35
    - (iv) On the Recognition of Sentence Types..........36
    - (v) Syntactic Processing..........38
5. Textual Macrostructure..........40

MAJOR ACTIVITIES AND ACCOMPLISHMENTS DURING REPORT PERIOD

1. "Cylinders," A Linguistic Data Structure Based on a Novel Use of "Rings" - P. Weston

A form of linked data structure called the "Halo" has been described elsewhere(1). The success of this concept suggested a further generalization of this structure and resulted in another more versatile structural concept, the "Cylinder," whose properties will be discussed in the following pages.

(i) Introduction. With the growth of non-numerical computer applications, and the attendant necessity of dealing with entities lacking the orderly internal relationships characteristic of matrices, interest in linked data structure has grown, with the key developments coming out of artificial intelligence research where indeed the data manipulation problems are the most severe.

Structuring of data with the computer store is a factor, implicit or explicit, in the design of any computer program. The point of tangency between data structure and program design lies in the necessity of imposing relationships upon data independently of their values, in order simply to express an algorithm or computing procedure upon them. This is particularly clearly seen in, for example, the case of matrix multiplication, where row and column relationships must be imposed in an appropriate way upon a set of stored numbers in order for the required procedure to be definable as a program. As a general principle, it may be said that, in the context of computation, data structure is dependent upon the processes to be performed, and is less an intrinsic quality of the data themselves.

In numerical applications it is ordinarily sufficient to structure data in matrix form provided that arrays with various numbers of dimensions can be handled, and compilers oriented toward numerical problems, such as FORTRAN and its sucessors, contain simple and direct means, including subscripting of variables and dimension declarations of arrays, for expressing the necessary relationships. What sets this type of structuring apart from linked structures is the fact that it is possible in an array to compute, in a reasonably simple manner, the location in the computer store--or address--which holds a given datum, on the basis of its description as an array element, i.e., the values of its subscripts and a "base address" corresponding to the variable itself. Indeed, even at the hardware level, current computers universally include means for dealing directly with one-dimensional structures through the use of address-modification registers.

It will be convenient to refer to all structures based solely upon computation of "addresses" as array structures. Linked structures on the other hand are characterized by stored rather than computed addresses, various means being used to explicitly store with the data the locations of related items. These stored "pointers" to other items of data are often also referred to as links, and structures built from them are often called lists or linked lists. While the linking method is theoretically completely general, in practice it involves an overhead cost in terms of storage space and running time and is used only when necessary.

When an algorithm effectively precludes the use of arrays, some form of linked structure is the only presently available alternative, since content-addressed stores--which are the other presently interesting possibility--are not now included in ordinary equipment and may never be. The difficulties which lead to abandonment of arrays are encountered when the details of the required structure are either; (a) highly irregular, leading to great quantities of wasted storage due to empty positions in an array, or (b) highly variable, leading to massive shuffling around of storage contents when array parameters are redefined. An important subcase of the latter is that in which structural requirements are substantially dependent upon data values, as happens in most instances of artificial intelligence problems. The automatic parsing of sentences can be taken as an illustration. The entire object of such an algorithm is the generation of structure upon the incoming data. In this case, the structure represents the syntactic relationships to be found in the input sentences, which are initially given simply as strings of symbols.

Because our work is primarily in the artificial intelligence area, the need of a good linked-structure system has been recognized as an important consideration. Two of the older and well established linked-list languages, IPL-V(2) and LISP(3) were not felt to be entirely adequate, due mainly to the restricted class of linkage structures realizable, i.e., in both cases only unidirectional linking is possible and no reentrant linkages are allowed, with the result that from an arbitrary internal element in a linked structure only a portion of the remaining structure is accessible. While for many purposes this objection is not actually as serious as it sounds, there do arise many cases in which complete availability of a structure from any element in it is a decided advantage. There have been attempts to provide more or less direct reversibility of linkage paths, e.g., Threaded Lists(4), and the SLIP system(5). The latter tends to impose heavy burdens upon memory space and the former involve some potentially severe problems due to untraceable connections when storage cells are "erased".

More recently, the usefulness of ring structures has been demonstrated(6), and they offer several advantages over the earlier approaches. The principal ones are (a) as the name suggests, the basic structures are closed upon themselves, so that complete availability of structure is implicit in the design without a great increase in storage space, and (b) as they have been implemented, ring structures usefully combine linking and array structuring and in so doing make possible a fundamentally more efficient use of the computer. A further, most significant advance has been the invention of general-purpose low-level linked-list programming languages, the most widely distributed one being "L6"(7). In sharp contrast with earlier approaches these are not tied to any particular linkage scheme, but are sufficiently general in concept and economical in the programming means they provide, to enable the user to realize whatever form of linked structure he may desire, using a fairly natural and compact programming notation.

(ii) Ring Structures. Since December, 1967, L6 has been available at CSL on the CDC 1604 computer and a cylinder system has been programmed using it. This system, which we call CYL6, is now in active use as a tool in other programming work.

In the terminology of linked lists computer storage is dividable into blocks and fields, in addition to the common word unit. A block is a group of consecutive computer words, i.e., having consecutive addresses, while a field comprises an arbitrarily defined region internal to a block, having limits fixed with respect to the block boundaries but not necessarily coinciding with any word boundaries. A common schematic representation for these relationships is shown in Figure 1. A linked structure may be created by using some fields in each block to hold the addresses of other blocks, i.e., to contain pointers. This is often represented in the way shown in Figure 2.

As a list structure a simple ring, which is the basic linked structure in the Cylinder system,is nothing more than a chain of pointers which closes upon itself. This can be represented as in Figure 3. Although the circularity of this type of structure could apparently lead a program to step endlessly from link to link in the course of a data search, (and for this type of reason all reentrant structures were avoided in early linked-list systems), it is in fact a simple matter to prevent such a difficulty, when the ring is the only closed linkage pattern allowed.

This is typically done by incorporating a specially designated and identifiable cell in each ring to serve as a reference point in searching and building functions. A similar device was used in our earlier Halo design, but in the Cylinder system emphasis is placed on eliminating all specially reserved storage for system functions--and most other restrictive system conventions--and the Cylinder ring is an entirely homogeneous structure. To insure termination of all ring-searching operations, the subroutines which execute them have been designed to

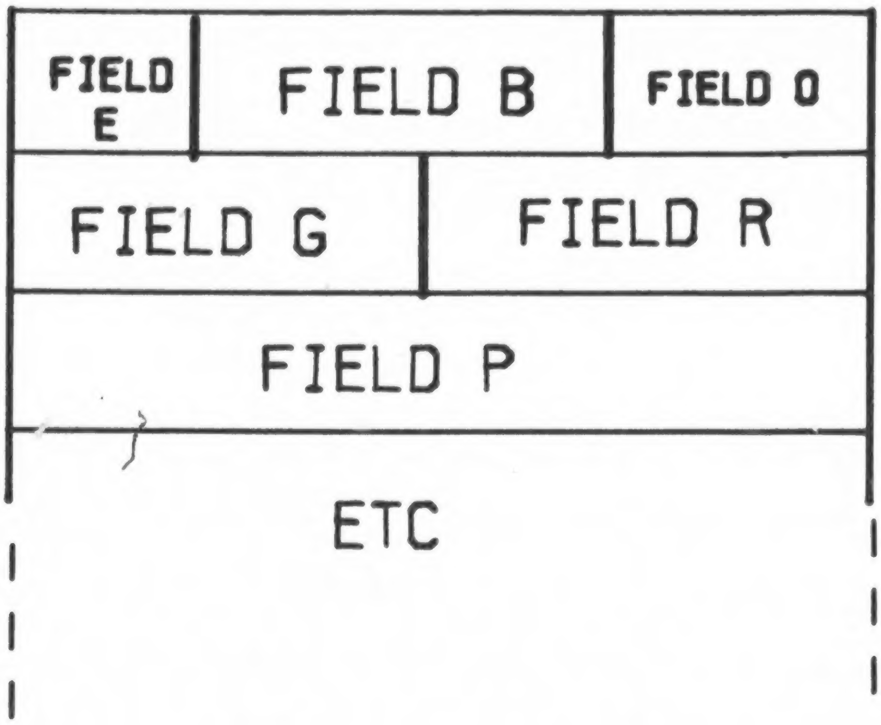

FIELD E
FIELD B
FIELD D
FIELD G
FIELD R
FIELD P
ETC

Figure 1. A block is a group of consecutive computer words, within which areas called fields may be arbitrarily defined and labeled.

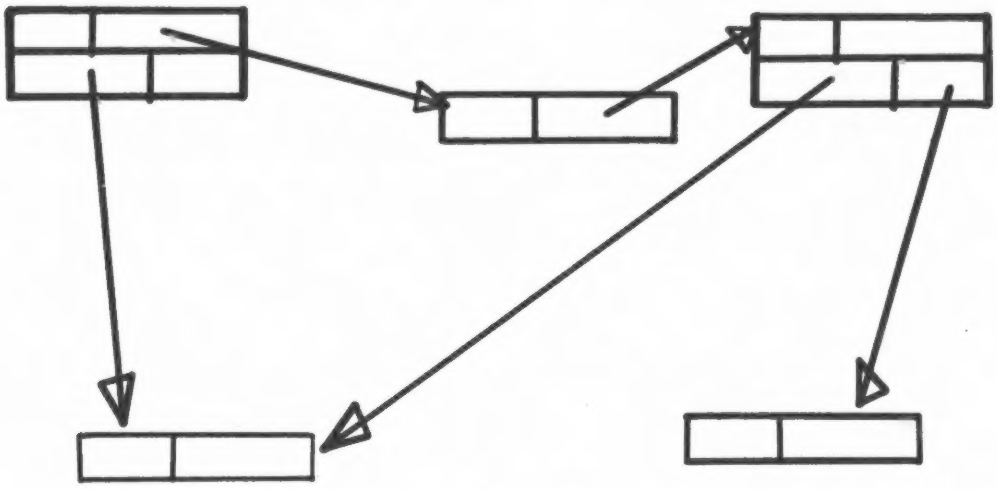


Figure 2. A schematic convention for linked structures, using arrows to indicate the blocks which are pointed to by stored links.

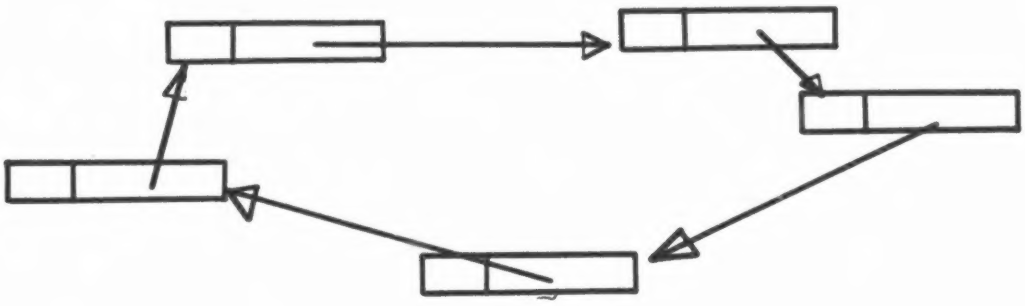


Figure 3. A simple ring is a closed chain of pointers.

automatically stop with a "fail" indication in the event that the starting point of the search is reencountered. This approach reduces storage overhead requirements and simplifies system conventions, simultaneously giving the user maximum latitude.

Given the choice of rings as the basic linkage pattern, which insures complete availability of each point in a structure from any other point with a minimally complicated set of system conventions, there still remains a wide field of choice in the exact manner in which the rings are to be used in representing data structure. Common to all methods is the use of rings to "thread" blocks of storage which may contain data as well as ring pointers, linking the blocks together somewhat as if they were beads on a string. One direct application of this basic scheme is in representing a classification of data. See Figure 4. Using individual blocks to hold descriptive information for the data,the representation is constructed by linking together on rings those data blocks which belong to each subclass to be represented. Because room can be made for more than one ring to thread a block, thereby representing membership of a datum in several classes simultaneously, there is no difficulty in setting up arbitrary hierarchical or overlapping classifications upon a set of data. Using ring structures in this way requires little or no more storage than conventional lists do, while in contrast with conventional lists, rings allow full flexibility in the manner in which data may be searched, i.e., with equal ease a class (ring) may be searched for the data (blocks) it contains, or a datum may be searched for the classes which contain it.

(iii) Cylinders. Further discussion of the use of rings in creating data structures will be limited to the context of the cylinder conventions. These are:

(a) The only allowed linked structure is the ring, i.e., a non-branching, self-closed chain of pointers.

(b) A fixed number of fields, eight in the case of CYL6, are defined upon the first words of all blocks, and all ring pointers within the system must be contained in just these fields. However, there are no system-imposed restrictions on the use of fields which are not needed in a given program for building rings, i.e., if only a few levels are needed short blocks may be used or data may be stored at the unused levels, wherever in the block they may occur. Program context is the determining factor. In CYL6 the eight ring-pointer fields are identified by the letters "I" through "P".

(c) All of the pointers constituting any one ring lie in the same field within their respective blocks, i.e., if one pointer in a ring is contained in a "K" field, all pointers in that ring necessarily lie in "K" fields. It is convenient to speak of the allowed pointer fields as

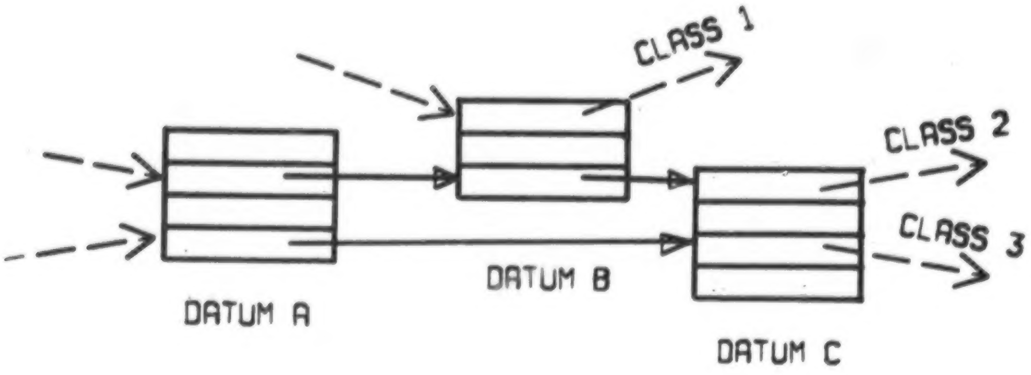

CLASS 1
CLASS 2
CLASS 3
DATUM A
DATUM B
DATUM C

Figure 4. Threading of blocks by multiple rings to represent a classification upon a set of data.

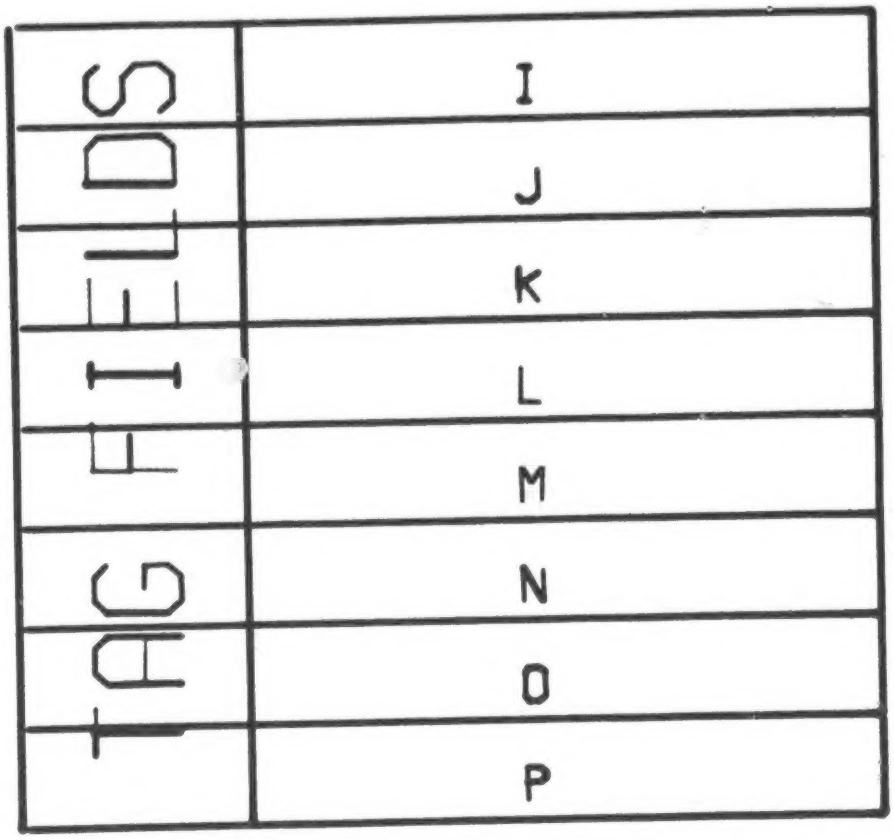

TAG FIELDS
I
J
K
L
M
N
O
P

Figure 5. Conceptual scheme of pointer levels as defined in the CYL6 system. This is not the actual arrangement in storage.

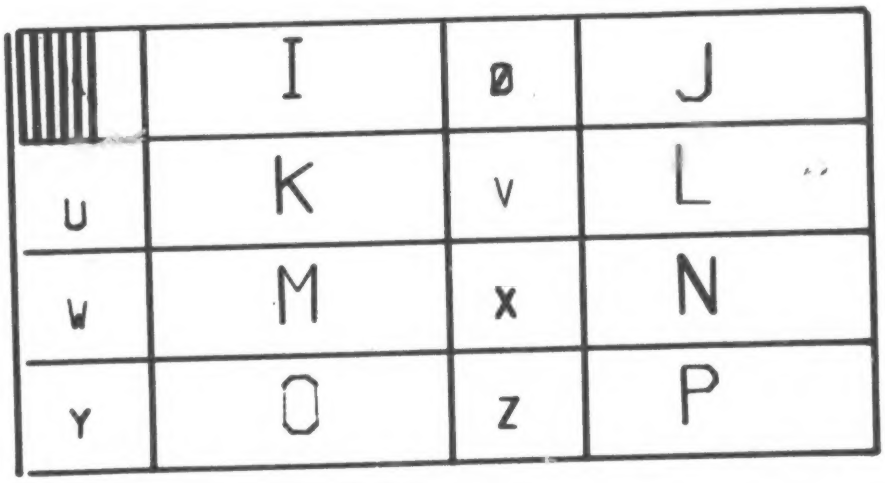

I
0
J
U
K
V
L
W
M
X
N
Y
O
Z
P

Figure 6. Actual storage arrangement in CYL6, using the first four words in a block. Field 0 is the tag field shared by levels I and J. Fields U through Z are the tag fields for levels K through P. The first six bits of the first word are reserved for the CSL7 storage allocator.

"levels" at which rings may be constructed and to speak of a ring as being at a certain level, e.g., in CYL6 a particular ring might be at the "K" level or "I" level, etc. See Figure 5.

(d) Associated with each pointer field there is also a data field of limited size, called a Tag field. All data search functions in the CYL6 system are sensitive only to the contents of the tag field associated with the level being searched. All tag contents are completely under control of the user and are primarily intended to identify, for search purposes, functionally distinct types of storage blocks within the context of an individual program. In many applications tags can provide all of the identifying information needed to govern every desired mode of search in a structure. In CYL6, the tags have a capacity of nine bits, which may be divided into independent three and six bit subfields or may be taken altogether for search purposes. It is convenient to describe the contents of a CYL6 tag field with an octal digit followed by a BCD character, e.g.,"5A", "1/".

A peculiarity of the CYL6 system is that while there are eight ring levels there are only seven tag fields, the I and J levels having a single tag field in common. This is due to interaction with space reserved in the first word of each block by the storage allocator of CSL7(8), a dialect of L6, the language in which CYL6 is written. See Figure 6.

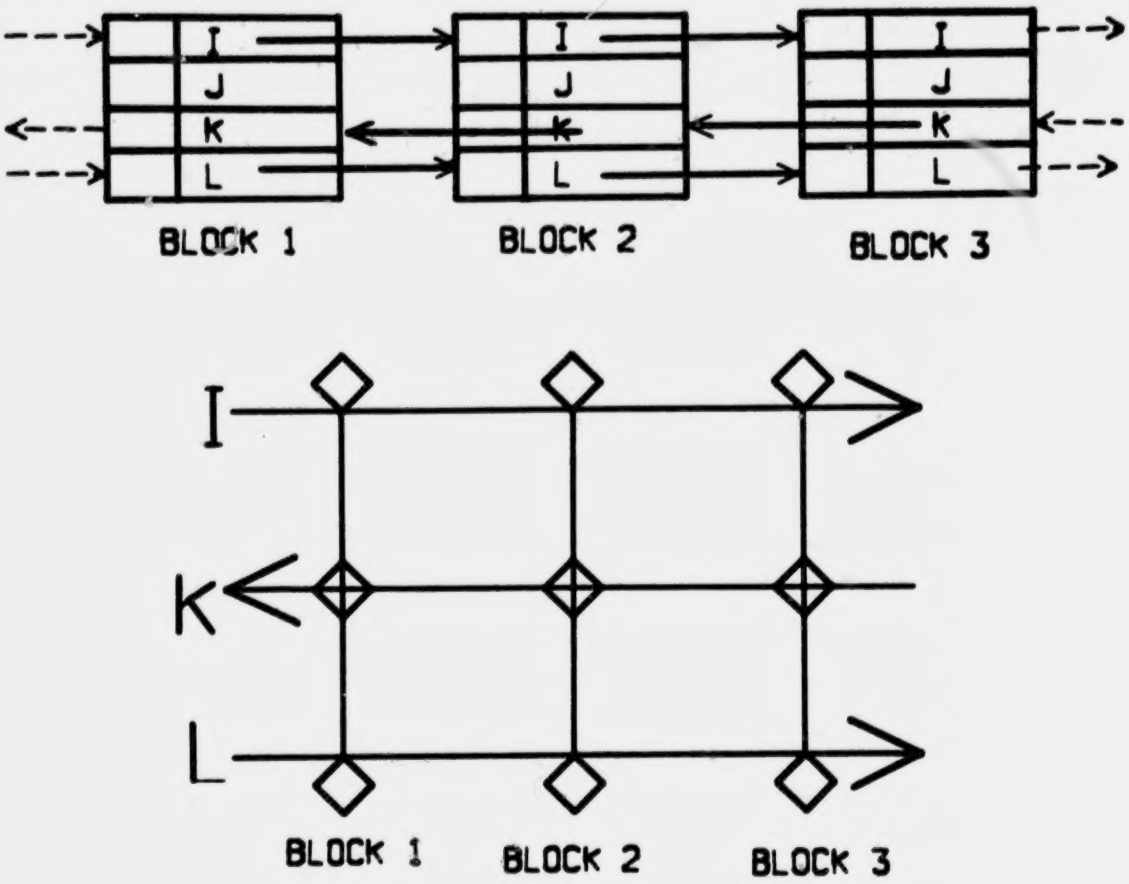

I
J
K
L
BLOCK 1
BLOCK 2
BLOCK 3
I
K
L
BLOCK 1
BLOCK 2
BLOCK 3

Figure 7. A schematic convention for cylinder structures. Above is the block and pointer equivalent of the lower diagram. The long arrows are understood to signify closed rings of pointers running in the sequence indicated by the arrow direction.

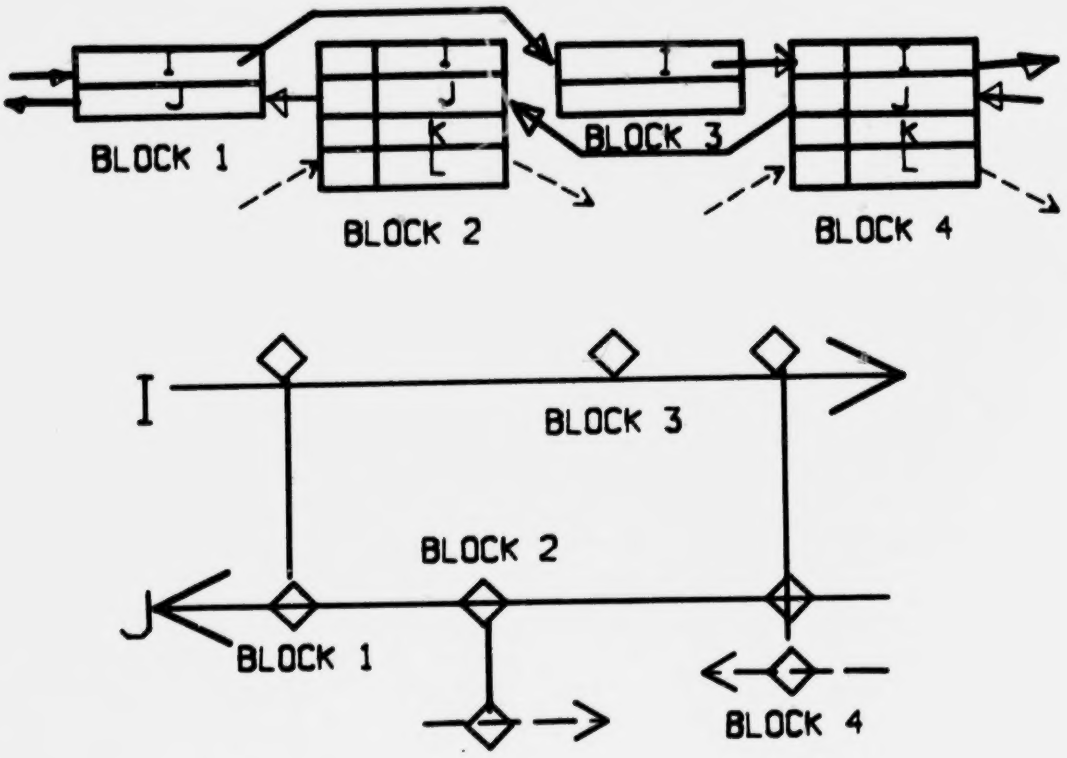

I
J
K
L
BLOCK 1
BLOCK 2
BLOCK 3
BLOCK 4
I
BLOCK 3
BLOCK 2
J
BLOCK 1
BLOCK 4

Figure 8. The basic linkages possible when a particular pair of rings is taken separately. The particular choice of levels here is unimportant.

The four conventions listed above are the only restrictions which are required by cylinders. They define a system broad enough in scope to include most other ring structures and, in the case of CYL6, which operates within a variant of L6, they are able to coexist with any other constructable in that language.

To gain further understanding of the structures that are available through the cylinder system it will be helpful to use an appropriate notation, such as the one which grew out of the CYL6 development effort. Figures 7 through 11 help to demonstrate this notation and its interpretation in terms of the block and pointer notation of the earlier figures. In this scheme the chain of pointers constituting a ring is represented simply by an arrow, the direction of which indicates the sequence of the indicated data elements along the ring. Since it is understood that all pointer chains are closed upon themselves, this fact may be kept in mind making it unnecessary to draw the rings as closed figures. When a single block is threaded by rings at several levels it is natural in the cylinder context to treat such a block as a cross-linking between rings and this is the way it is drawn, as in Figure 7.

Although it is in fact generally the case that a structure consisting simply of crosslinked rings will tend to be unwieldly in any notation, some order can be extracted from the chaos by limiting one's attention to a single pair of rings at a time. In this case the possibilities are quite limited, a block is threaded by none, one, or both of the rings, and in the latter two cases may or may not be threaded by other rings as well, Figure 8.

The basic cylinder is formed from a pair of rings and employs cross linking between them in a particular way. The framework of the scheme is pictured in Figure 9, where the triangles are used to indicate the positions on each ring of the special cross-links which we have called "seams". The only defining property of the seam substructure is that the blocks which form the seams lie either in exactly identical or more usually in exactly reverse sequences on the two rings. In the figure the arrow directions shown indicate this ordering. The existence of such seams is not a mandatory feature in the CYL6 system. Their insertion, removal, and selective tagging is left entirely up to the user's discretion.

The usefulness of having a sequence of identifiable seam cross links with the indicated ordering lies in the fact that between each pair of seams a type of subring is created which has the important characteristic of self-closure as does the main ring but has several added features of some usefulness as well. As Figure 9 shows, each subring contains four segments, consisting of the two sequences of ring links lying between the two bounding seams at the lower and upper levels respectively, and of the two seams themselves, which constitute bilateral connections between the upper and lower ring segments. One natural way of using this ring and subring configuration to represent data structure is to assign a datum to each seam and to represent binary relations between data by other cross links which carry tags identifying the relations they represent and are threaded by the lower ring segment of one subring, whose seam represents the value of the first argument of the relation and by the upper ring segment of the subring for the datum at the second argument position. Such a representation is shown in Figure 9.

This representation for binary relations is highly compact as a linked structure, and is quite complete in representational power in view of the facts that both the forward and inverse relation are simultaneously represented, and that the main rings constitute a master file from which all the data or all the links representing any particular relation may be selectively retrieved using a tag-controlled data search. The storage requirement in terms of number of necessary links compares quite favorably in this case with that for IPL-V,provided that all the above information is to be represented. In the case of IPL-V there is a basic requirement of two links per datum, plus eight links to represent both a binary

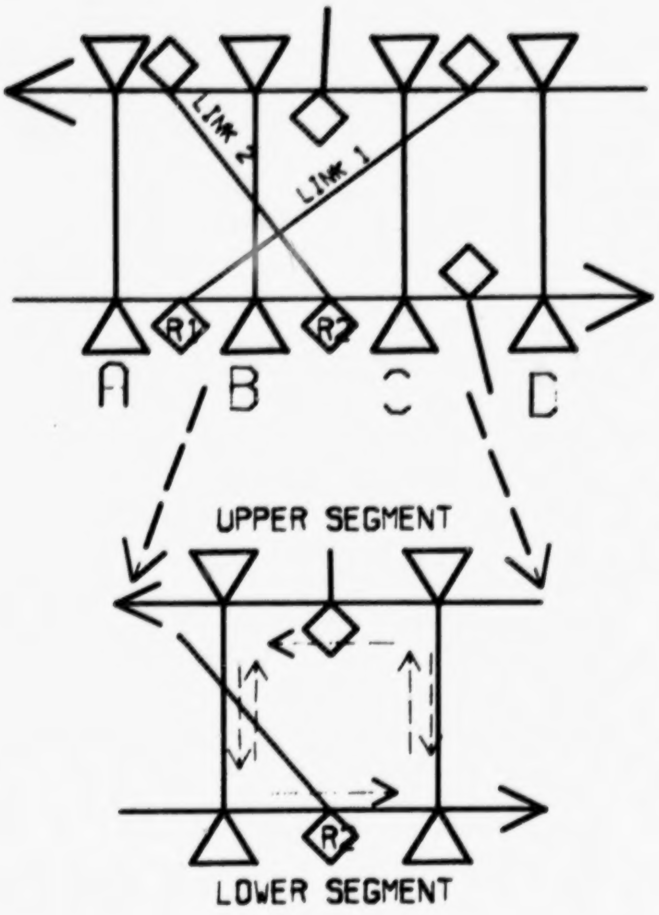

LINK 2
LINK 1
R1
R2
A
B
C
D
UPPER SEGMENT
R2
LOWER SEGMENT

Figure 9. The subring structure based on 'seams'. In the upper structure links 1 and 2 represent R1(a,c) and R2(b,a) respectively according to the binary relational scheme described in the text.

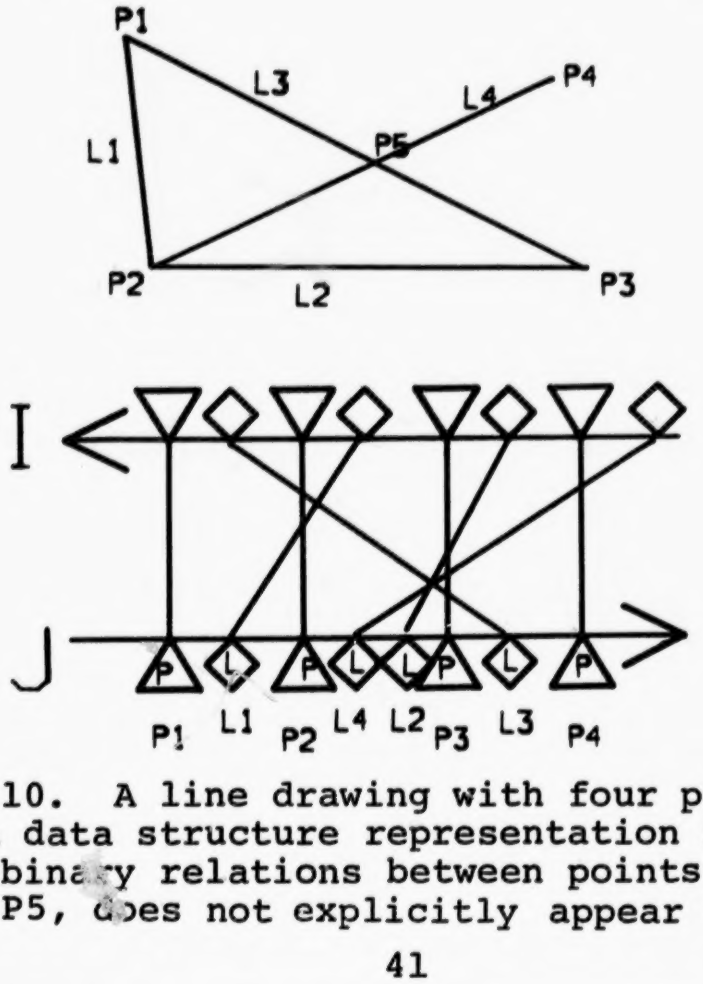

P1
L3
L4
P4
L1
P5
P2
L2
P3
I
J
P
L
P
L
L
P
L
P
P1
L1
P2
L4
L2
P3
L3
P4

Figure 10. A line drawing with four points and four lines, and a data structure representation for it in terms of lines as binary relations between points. The intersection point, P5, does not explicitly appear in this structure.

relation and its inverse, plus at least two per datum to construct the master file, plus two for each computer word of data (numbers, alphanumeric strings, etc.) required to represent the contents of each datum, making a total for one binary link between two data, each containing two computer words, of twenty-four links. Using Cylinders, the requirement is two and one-half links for each datum, counting a tag field as one-half of a link field, two and one-half for each binary relationship represented, and no extra links for the contents of a datum because of the block structure. The total requirement for the case cited above is seven and one-half links, which is less than one third of the IPL-V requirement and the comparison improves as more words are needed for each datum.

(iv) Examples of Use. An example of a binary relational structure of a rather simple kind arises in the machine representation of line drawings involving points and straight lines as elements. If points are taken as the data, then lines can be represented as binary relations between the data points, and a structure of the sort depicted in Figure 10 results. This structure represents only the connections of the points and can stand for any pattern consisting of a triangle with a fourth line ending on one of its vertices. Supplying coordinate values for the points realizes a particular example.

As an example of the flexibility of cylinders, Figure 11 shows a completely compatible extension of the scheme in Figure 10 which allows the simultaneous representation of line to line relations, such as intersection, and of point-to-point relations including the line definitions in terms of point pairs. At the K and L levels a second cylinder is formed, having as seams the lines which appear as relational links at the upper level. Any points which are defined by line intersections, such as point 5 in the diagram of Figure 10, can appear as relation links at the lower level but as seams at the upper level, thus becoming capable of serving to define further lines and so on.

Because points which arise from line intersections depend for their coordinates upon the other points which define the intersecting lines, this dependence is important to any program which operates upon such data structures and must be somehow marked, without destroying the identifiability of the datum as a point. This is readily done through the use of say, the upper subfield of the tag, in CYL6 marking it distinctly from the corresponding subfield for other points but keeping the lower portion the same. In this way the tags themselves can be made to represent a type of classification upon the data with no further burden upon the structure or storage requirement.

(v) Summary. A data structure concept which is an adaptation of ring structures has been developed and described. It is highly compact compared with conventional

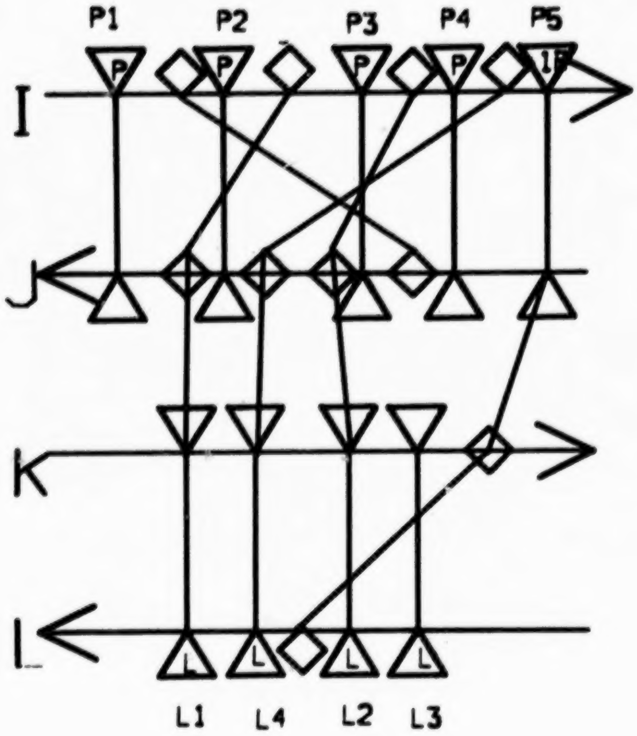


Figure 11. An extension of the scheme of Figure 10 which allows intersection points to be represented as binary relations between lines in a cylinder at one level while the same lines are simultaneously represented as relations between points in a cylinder at another level. P5 is given a distinct tag value, lp, because it must be distinguished as a dependent quantity, and is not a freely variable point.

lists, possessing an advantage of typically a factor of three in storage requirements, and it is highly flexible in use, since its definition includes most other ring systems, but the elimination of almost all unnecessary system conventions, and a novel employment of the ring structure tend to give Cylinders an advantage in compactness and flexibility in this field as well. It is now being actively employed in programming for artificial intelligence and its further potential thereby being explored.

REFERENCES

1. Weston, P., "Data Structures for Computations within Networks of Relations", in BCL Report 67.2, Biological Computer Laboratory, University of Illinois, Urbana, 126 pp., (1967).
2. Newell, A. et al., Information Processing Language - V, Prentice-Hall, Englewood Cliffs, N. J., (1964).
3. McCarthy, et al., LISP 1.5 Programmer's Manual, MIT Press, Cambridge, Mass., (1962).
4. Perlis, A. J. and C. Thornton, "Symbol Manipulation by Threaded Lists", CACM 3, 195-204, (1960).
5. Weizenbaum, J., "Symmetric List Processor", CACM 6 524-543, (1963).
6. Roberts, L. G., "Graphical Communication and Control Languages" in Second Congress on Information System Sciences, Hot Springs, Virginia, (1964).
7. Knowlton, K. C., "A Programmer's Description of L6", CACM 9, 616-625, (1966).
8. Bouknight, J., Preliminary User's Manual CSL6 CSL7, Coordinated Science Laboratory, University of Illinois, Urbana, (1967).

2 An Objective Function for the Scheduling Routines of a Time-Sharing System - H. Bielowski

In order to be economically feasible, conversational interaction with large information processing systems makes it necessary to apply the principle of "time-sharing".

For many problems associated with time-sharing, such as virtual memory techniques, memory protections, etc., a fairly satisfactory solution has been found, whereas poor solutions of scheduling problems are responsible in many cases for the typical low efficiency and frustrating performance of the more general-purpose systems.

In the course of a literature study, it became apparent that the following reasons must be held responsible for this situation:

1. There is no consensus on a suitable objective function for the scheduling of programs.

2. There is no a-priori information available to the scheduler on the characteristics of the "jobs" which are called for by the users.

3. The various scheduling processes in a given system do not interact sensibly.

4. The system behavior is tailored according to assumptions on the job mix and user behavior which do not

reflect reality.

5. An optimal mathematical solution of many scheduling problems is not feasible. Even many of the suboptimal solution procedures are computationally very involved.

With respect to the first point regarding the lack of an objective scheduling function, a criterion has been developed that seems to enable the scheduler to adjust the behavior of the system to arbitrary job streams.

The obvious goal is the satisfaction of the users as a whole, or what should be equivalent, the best possible revenue R_τ of a computer installation for a given period of time τ.

For each request of job i, a "revenue function" $R_i(t_i)$ can be constructed, which reflects contributions to R_τ independent of the response time t_i the system realizes.

The revenue criterion presents itself now as: "Maximize $R_\tau = \sum_i R_i(t_i)$ for all jobs i in period τ." By their definition, revenue functions contain estimates. Therefore, high precision in their representation is not required. Furthermore, scheduling economy will demand their simplification very strongly.

These functions are characterized by the customer's importance, the fee he pays for a job of a given size and kind, the job size (≃ standard response time), and the kind of urgency that is attributed to his job. The functions can be "constructed" by composing information entered by the customer together with his request, or stored in his terminal, or stored in job-standards tables in the system. The entries in the job-standards tables can be updated by the manager of the computing center, using a special routine.

We discuss now linear functions (see Figure 12). They are represented on a real time axis with arbitrary origin,

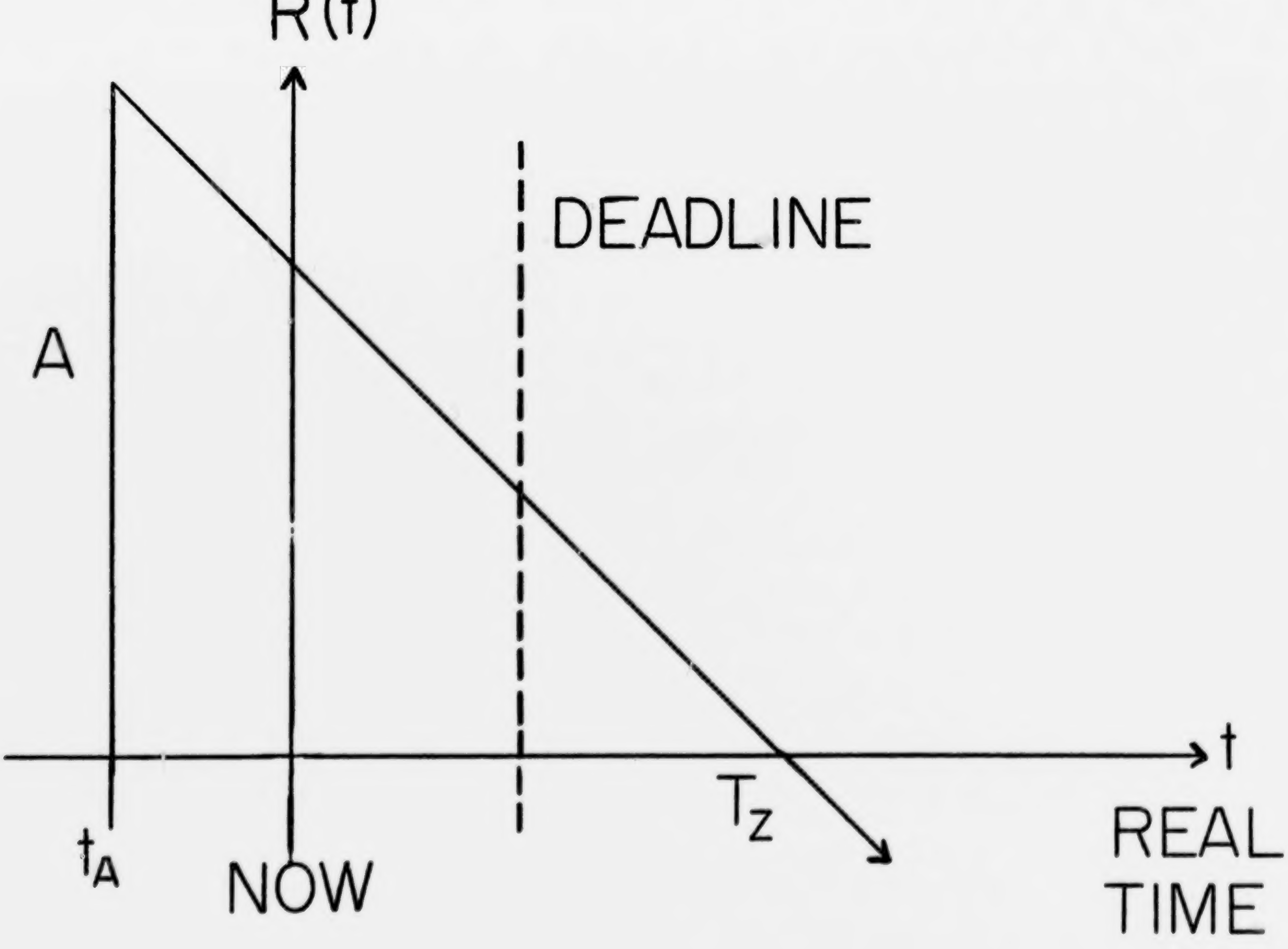


Figure 12. A linear revenue function

as they actually have to be seen by the scheduler. The linear functions for each job i are given by the arrival time t_{Ai}, the "zero revenue response time" T_{Zi} and the maximum possible revenue A_i, which is realized only at instant response. In addition, the "standard response time" T_i has to be given. T_i should ideally be the time it takes to finish the job. As this is not known exactly in advance, and depends of course on the competitive situation (competition for the use of hardware units, i.e., on the length of the I/O-queues), some standard must be adopted, which could be corrected by the scheduler according to the system state, and shortened for work in progress.

It should be noted, that T_Z will not be equal to the deadline, because a response at deadline time is still satisfactory. The negative part of the function is necessary to represent loss of future revenue, etc., because of user chagrin.

In the simple case, where the scheduler faces only two jobs in a queue, (see Figure 13), the following analysis generates the relative priorities:

In real time t, [capital T indicating $(t-t_A)$],

$R_1(t) = A_1-(A_1/T_{Z1})(t-t_{A1})$ and $R_2(t) = A_2-(A_2/T_{Z2})(t-t_{A2})$

There are two alternatives:

$$\text{job 1 first:} \quad R^1 = R_1(t+T_1)+R_2(t+T_1+T_2)$$

$$\text{job 2 first:} \quad R^2 = R_1(t+T_1+T_2)+R_2(t+T_2)$$

The first alternative would be chosen, if $R^1>R^2$. This is true for $A_1/T_{Z1}T_1 < A_2/T_{Z2}T_2$.

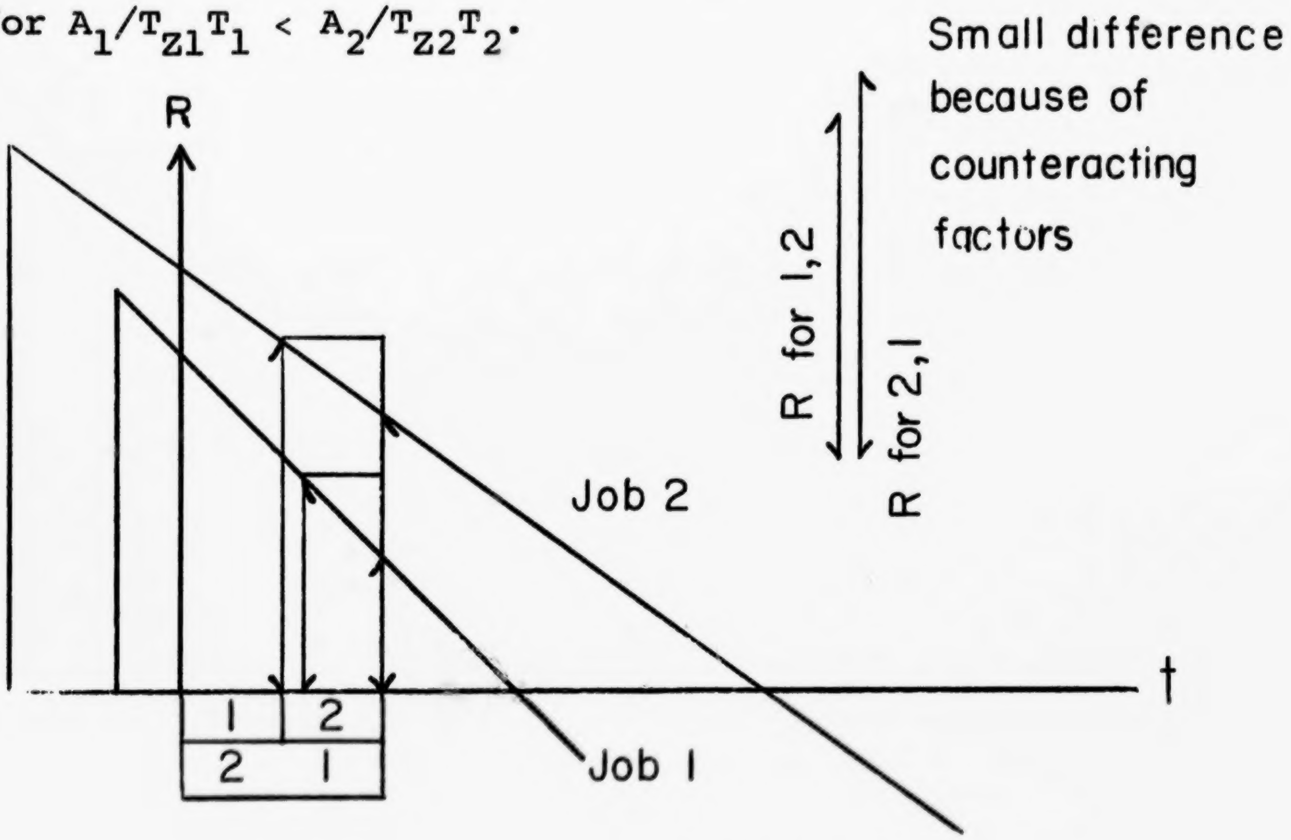

Small difference
because of
counteracting
factors
R
R for 1,2
R for 2,1
Job 2
Job 1
t
1
2
2
1

Figure 13. Linear revenue functions for two jobs and the revenue of the two alternative service sequences (no time--slicing assumed).

It is interesting to note, that the arrival times t_{Ai} cancel out and do not appear in the decision rule. That is because the functions are unlimited and decrease indefinitely with time. Obviously, this does not hold for reality. After a certain time, it does not make any difference whether a job is executed or not. The revenue function will simply continue as a negative constant, or cease to exist, but expresses the fact, that the job is rejected. This brings to mind, that it might be of more value to the user, if the job were rejected before it started. What is important is the fact, that the decision rule only has this form, as long as the sum of the standard response times T_i of all jobs in the system (adjusted for work in progress) is smaller than the remaining time span before the next "stop point" occurs. Even if we add the notion of the "stop point", the linear function does not satisfy our intuition of what these revenue functions really look like. There will certainly be a lower bound on the response time, below which the revenue will not be notably higher, which appears to correspond to the conventional concept of a deadline. Therefore, consider functions of the type given by Figure 14 (on next page) as representative of the general case.

For more than two jobs in a queue, say N, then all N! possible sequences should theoretically be evaluated. For longer queues, this enumerative method is of course unfeasible. Heuristic suboptimal approaches have to be chosen. It appears to be sufficient to decide, for instance, on which of N possible locations a new job must be placed in a queue which is served strictly in a "first-out"

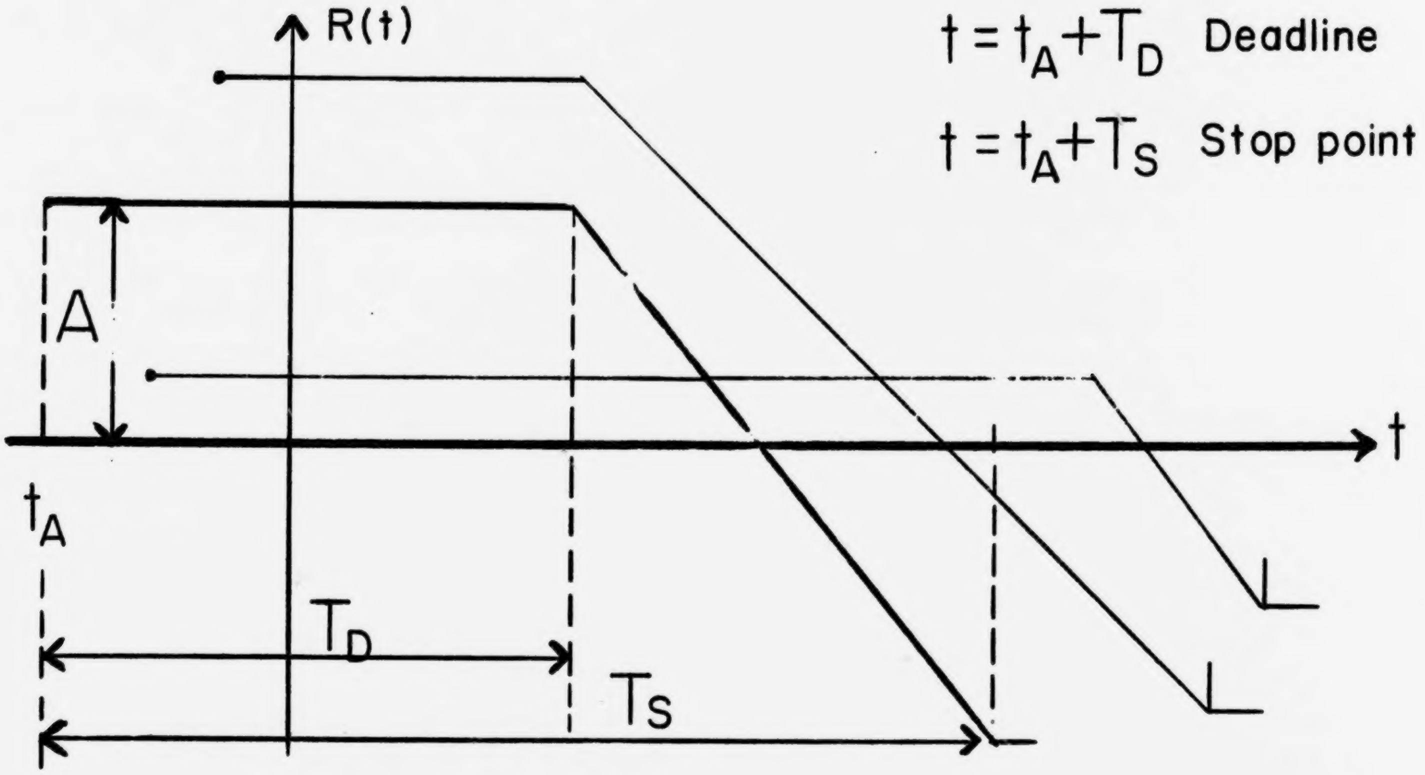


Figure 14. Some examples for the class of revenue functions which is considered to be representative for most jobs.

manner. This type of analysis would make it possible to devise a set of priority rules (to be used by the scheduler) corresponding to classes of revenue functions (combinations of T_D, T_S and A) and values of T of both the jobs waiting and the jobs in process.

REFERENCES

1. Belady, L. A., "A Study of Replacement Algorithms for a Virtual Storage Computer," IBM Systems Journal, 5, 2, (1966).

2. Denning, D. J., "Effects of Scheduling on File Memory Operations," Proc. of the SJCC, AFIPS, (1967).

3. Estrin, G., and L. Kleinrock, "Measures, Models and Measurements for Time-Shared Computer Utilities," Proc. of the 22nd. National Conference of the ACM, (1967).

4. Fisher, R. O., and C. D. Shepard, "Time-Sharing on a Computer with a Small Memory," CACM, 10, 2, (1967).

5. Greenberger, M., "The Priority Problem and Computer Time-Sharing," Management Science, 12, 11, (1966).

6. Kleinrock, L., "Time-Shared Systems, a Theoretical Treatment," JACM 4, 2, (1967).

7. Manacher, G. K., "Production and Stabilization of Real Time Task Schedules," JACM, 14, 3, (1967).

8. Nielsen, N. R., "The Simulation of Time-Sharing Systems," CACM, 10, 7, (1967).

9. Rosene, A., "Memory Allocation for Multiprocessors," IEEE Transactions on Electronic Computation, EC-16, 5, (1967).

10. Sherr, A. L., An Analysis of Time-Shared Computer Systems, MAC-TR-18, Massachusetts Institute of Technology, Cambridge, (1965).

11. Smith, A. A., I/O in Time-Shared, Segmented Multiprocessor Systems, MAC-TR-28, Massachusetts Institute of Technology, Cambridge, (1966).

12. Vyssotsky, V. A., Corbato, F. J., and R. M. Graham, "Structure of the MULTICS Supervisor," Proc. of the FJCC, AFIPS, (1965).

3. Cognitive Memory - S.R. Ray, B. Wang, J. Chow

We have decided to forego, temporarily, the study of machine organizations suited for (an assumed form of) cognitive structure manipulation in favor of a more concentrated study of the fundamental interactions required within these data structures. Consequently, this reporting period has been consumed largely in attempting to specify a universe (of objects, attributes, and relations) in which some interesting features of cognition can be studied. One such feature, a hallmark of cognition, is the ability to calculate in terms of imprecise categories.

The choice of a universe and a detailed specification of the class of operations to be studied is not yet resolved.

4. A Pilot Information System: "Rules of the Road" - K. Biss, R.T. Chien, C. Hartmann, K. Kelley, F.P. Preparata, P. Reynolds, F. Stahl

(i) Introduction

R.T. Chien

The "Rules of the Road" is the official driver's manual for the State of Illinois. The goal of this sub-project is an information system based on knowledge contained in the document. The system will operate in the on-line mode and will answer reasonable questions concerning this data base. The purpose of this sub-project is many-fold:

1. It provides us with testing material for the effectiveness and efficiency of language-processing techniques (syntactic and semantic) developed here and elsewhere.
2. It provides a natural data base for the study of relational structure for information representation.
3. It provides a data base for studying the question-answering process with regard to both the logical aspects and the aspects related to organization, search, synthesis, and retrieval.

Although not the primary goal, it is expected that the system will be able to pass the written portion of the driver's test.

(ii) Preprocessing

F. Stahl

The entire text of "Rules of the Road," the driver's manual for the State of Illinois, has been keypunched for use as the data base for the question-answering fact retrieval system.

Also, a number of programs have been implemented to facilitate the manipulation of this data during the initial phase. They include a text-editing routine and a coordinate-indexing routine similar to IBM's KWIC index.

The text-editing program which uses the scope for display purposes and the lightpen for communication has been written by V. Metze to allow the correction of prose material. The program allows the user to specify delete, replace, or insert operations which act on the lines, words, or characters that he designates with the lightpen. Information inserted may be longer or shorter than the information deleted; the program readjusts the lines accordingly. Information is carried over from one line to the next as necessary. (However, the carry-over ends when a new paragraph is encountered.) Words are never broken in the middle as information is moved from one line to the next. The program contains a general string-mover which makes overlapping moves of information described by 18-bit addresses right or left in memory and correctly hooks on to the beginning and end words so as to retain any information not included in the move.

The text-editing routine will speed up the turn-around time on this manipulation of the data page in other phases of the operation.

The KWIC index has been used on the data base to get an estimate of the scope of the vocabulary and phrase usage. The inherent value of the index is in its ability to find all the occurrences of any word or phrase in the data base with a minimum amount of effort. This is being used as an aid for sentence classification and recognition of sentence types. In KWIC the context of a word is with respect to its line. A modified version is being implemented to make the context of a word a sentence. The modified KWIC will be used to extract phrases in a rigorous fashion from the data base. Words and phrases arrived at in this manner will be compiled into a dicitionary or thesaurus.

The dictionary will serve as a replacement structure to reduce the total vocabulary of the data base and put it in a more standardized form. It is hoped that this same dictionary can be used as the replacement structure for questions in the same manner as for the data base.

Work by Bobrow and Rapheal has suggested that if a given question has associated with it a small number of statements such that the answer to the question can be inferred from those statements, then there is no reason to presume that a machine cannot be organized to answer that question. We propose that given a question and a large number of statements, there are methods to eliminate large numbers of irrelevant statements and thereby reduce the problem to manageable size. This might be accomplished by a kind of intersection of relevant phrases as defined by the question and the linkages of those phrases in the text itself.

(iii) Steps Towards a Relational Structure

F.P. Preparata, K. Kelley, P. Reynolds

As was pointed out in the introduction, the body of knowledge on which the project is based should be represented as a relational structure.

The "universe" of vehicular road traffic should be adequately described through its items and their interrelations. It is felt that this representation should consist of two main portions: 1) A strictly relational component in which the essential elements of the universe are identified, listed and defined exclusively through their relations; 2) A propositional component, in which specific propositional functions, such as "legality," "safety," and the like are evaluated for events constructed in terms of elements described in the relational component.

A preliminary attempt has been made to construct the relational component. Three basic classes of items have been identified, that is, paths, vehicles and signs. Within the class of paths, the useful relations are rules of interconnection; within the other two classes, useful relations are generally of classificatory type (such as regulatory signs, informative signs, etc.). Interclass relations are generally of relative position or motion.

Since the relational component will find its primary application in the process of query interpretation (described in section D) its construction will proceed in parallel with the latter activity. It is felt that the findings of the study of sentence recognition will have a basic influence in directing the construction of the relational component.

Study of the propositional component will begin once sufficient insight is gained into the essential features of the relational component.

(iv) On the Recognition of Sentence Types

P. Reynolds, C. Hartmann

In a retrieval system an important parameter to be considered is the searching time. The classification of questions in types can greatly help to cut down the search time since the search can be guided according to the type of the question. The answers will be obtained by a matching with the data or by a theorem-proving technique if the data do not contain explicitly the answers.

In our particular case the subject is "Rules of the Road" and it is assumed that the computer has a minimum "knowledge" of the subject to recognize words and their grammatical categories which are related to the matter.

Many attempts to classify the questions were made but only the most promising will be described below.

Since only 60% of the questions begin with one of the following words: when, what, where, who, whom, which, why, how, how much; at first the sentences are classified into two groups. The first group includes sentences which begin with one of the described words and the second group includes sentences which do not. We believe that with a sentence inversion routine many of the questions in the second group can be put in the form of the first group.

The procedure for all sentences of the first group are very similar. We will illustrate the procedure by considering sentences beginning with "WHAT." The sentence is scanned only by looking at nouns and verbs which are not in clauses, and auxiliary verbs such as may, shall, will, must, should, would and can. The question will follow in one of the following patterns:

1. What N V S
2. What V_1 N V_2 S
3. What be (V) S
4. What AV N (V) S

where N is a noun, V, a verb, AV an auxiliary verb, an S is the set of statements (properties, relations, etc.) which will characterize the answer. In other words, for pattern 1, the answer will be the intersection of all things equal to S (modulo V) with N. Any ambiguity in the question will show up at this step. We have already an algorithm to classify the "WHAT" sentences. For example, after the application of the algorithm to the following sentence, "What markings are used for the center line of a four-lane highway?" we get the pattern N V S where N=markings, V=are used and S=for the center line of a four-lane highway.

The next step will be the classification of the part S, but S will not be classified word by word as we did until now; it will be by matching of an entirely new idea expressed in S (properties, relations, etc.). A sentence inversion routine will be developed to bring as many sentences as possible to the first group. The sentences of the second group will be considered last and some of them will be treated as special cases if necessary. In the meantime, the algorithms will be tested against the data to test their performances.

(v) Syntactic Processing

K. Biss

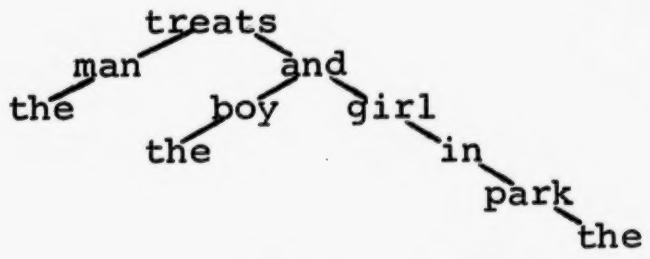
In the system we are developing to answer questions from the Illinois driver's training manual, "Rules of the Road," we will search for answers to questions on two levels.

On the first level we will have some sort of semantic matching between questions and text to pick out a set of sentences which will most probably answer the question. On the second level we will syntactically match the question with the reduced set of sentences from level one to pick the answer to the question.

Since we will have two levels of search in our system, the syntactic level need not be as powerful as in some other systems. For this reason we are seeking a simple, fast, efficient parsing algorithm for the second level.

Our search for an appropriate parsing algorithm has led us to a study of dependency grammars. In a dependency grammar each word of a sentence except a head word (usually the main verb) depends on some other word in the sentence. We say that one word depends on another if the first modifies or complements the meaning of the second. The head word of the sentence has dependents but is dependent on no other words in the sentence.

For example, the sentence, "The man treats the boy and the girl in the park," has the following structure:



McConlogue and Simmons[1] wrote a program to parse English sentences using a dependency grammar. They assigned numbers to the levels of the dependency tree and noted that if a particular word appeared at a certain level in one sentence, it would probably occur at the same level in other sentences.

Using the idea of assigning numbers to levels in the dependency tree we have parsed some sentences from "Rules of the Road."

It is hoped that with the aid of these parsed sentences we can construct a good parsing algorithm, probably with some learning features.

5. Textual Macrostructure - D.E. Carroll

During the present report period, work has continued along the lines previously indicated in the effort to explore certain underlying classificatory structures and relational processes that are conjectured to be significantly involved in the cognition and synthesis of documentary or textual information. The level of this effort is necessarily still a rudimentary one and remains centered presently on certain of the problems encountered in attempting the synthesis of differing accounts of a topic so as to yield a single account characterized by logical clarity and low redundancy. At present, work is proceeding primarily with the synthesis of differing accounts of selected topics in the field of document indexing and retrieval systems.

---

[1] Keren McConlogue and R. F. Simmons, "Analyzing English Syntax with a Pattern-Learning Parser," Comm. ACM, 8 (Nov. 1965) 687-698.

IV. ACCOMPLISHMENTS FROM 8/1/68 - 11/30/68

Preface

The following pages give a brief account of the activity associated with the study on cognitive memory during the fourth report period from 1 June, 1968 to 31 August, 1968. Technical papers that grew out of this study and were presented at scientific meetings, were published in journals or books, or have been published as technical reports are not reprinted in this Technical Progress Report. Reference to these papers are given in paragraph (vii) of Item 1.

Again, the contribution of members of the Department of Linguistics, of Mathematics, of Anthropology and of Psychology, whose association with this project is by interest and enthusiasm, rather than by contract, is herewith acknowledged with great gratitude and appreciation.

Last but not least I wish to express my thanks to Dr. John Lilly of the Communications Research Institute in Miami, Florida, who joined the group as Visiting Professor in the last weeks of July and whose important findings on phonetic and semantic alternates in the perception of repeated words constituted a significant stimulus to this project, as well as to Professor E. J. Scott from our Department of Mathematics who developed during the summer semester a most valuable formalization of various aspects in the chain of cognitive processes.

H.V.F.

ACCOMPLISHMENTS FROM 8/1/68 - 11/30/68

Table of Contents

Page

Preface..1

Major Activities and Accomplishments During Report Period.......3

1. A Pilot Information System R2: "Rules of the Road".........3

    (i) Introduction.......................................3
    (ii) Query Classification..............................3
    (iii) Syntactic Analysis...............................10
    (iv) Context Modeling..................................20
    (v) Concept Processing.................................24

2. Grammars and Relational Structure.......................27

3. Investigation of Fundamentals of Nonlinguistic Cognition...31

4. Cognition and Heuristics................................34

5. Machine Architecture for Information Retrieval..........36

    (i) Introduction.......................................36
    (ii) Investigation of the Processor of Lee and Paull.......38
    (iii) A New Associative Memory Processor..............44

6. Studies of the Mathematical Theory of Cognition..........49

    (i) On the Forms of Equations Associated with Inductive Inference Computers..........49
    (ii) On a Class of Nonlinear Property Filters.........53

MAJOR ACTIVITIES AND ACCOMPLISHMENTS DURING REPORT PERIOD

1. A Pilot Information System R2: "Rules of the Road" -

L. Biss, R. T. Chien, C. Hartmann, J. Jansen, D. Lombardi, F. P. Preparata, P. Reynolds, J. Schultz, F. Stahl

(i) Introduction

R. T. Chien

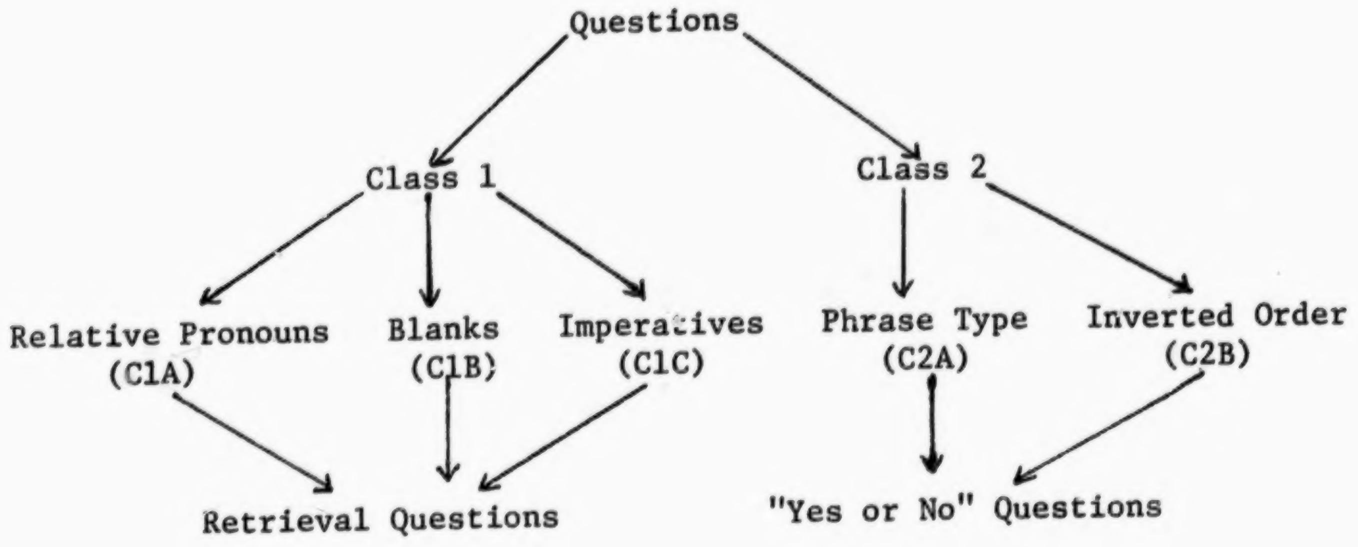
The R2 system is a general retrieval system based on the "Rules of the Road" manual of the State of Illinois. The objectives of the R2 system program are twofold. First, the system is to provide us with a testing facility and data for the development and evolution of fundamental concepts in cognition. Second, it is to provide us with a general question-answering system with significantly broader scope than systems presently known.

During the previous period, a number of significant achievements have been made in the areas of relational structuring, parsing programs, context-modeling, and query analysis. The details of these results are summarized in the following sections.

(ii) Query Classification

P. Reynolds, C. Hartmann, R. T. Chien

The key step in a question-answering system is to understand what the question means. When the question is understood, its meaning can be processed with a retrieval system, which will return an answer to the question. Our problem has been how to determine at least partially what the question means by normalization. It is found that we could normalize all questions in a step-wise manner as given by the following diagram:


Questions
Class 1
Class 2
Relative Pronouns (C1A)
Blanks (C1B)
Imperatives (C1C)
Phrase Type (C2A)
Inverted Order (C2B)
Retrieval Questions
"Yes or No" Questions

First, we divided the questions into two classes. Class 1 questions would typically be answered by a sentence containing some retrieved information, while Class 2 questions would be answered by "yes, no, true, false." Class 2 sentences could be immediately recognized by the presence of such phrases (C2A) as "yes or no," "true or false," or by the presence of an inverted order, subject-predicate structure (C2B). The inverted order, subject-predicate structure, can be recognized in all cases by the presence of an auxiliary verb (must, can, is, does, etc.) at the beginning of the sentence or after a comma in the sentence, e.g.,

"On an interstate highway, can a car travel at 40 mph?"

"Must you have automobile insurance in order to drive?"

Class 1 consists of all other questions. It is characterized by the presence of a relative pronoun (C1A), a blank (C1B), or an imperative (ClC). Class ClC is recognized by the presence of a word such as "list, give, state" at the beginning of a sentence or after a comma, e.g.,

> "State the difference between a car and a motorcycle."

Phrases such as "would you please" in

> "Would you please state the difference between a car and a motorcycle?"

are disregarded.

Class ClB sentences are "fill-in the blank type," e.g.,

> "An octagonal-shaped sign means _______."

They are recognized by the presence of a blank.

Class ClA questions always contain a relative pronoun such as "what, when, where, who, whom, which, why, how, how much, how many," e.g.,

> "What is the minimum speed for a car on an interstate highway?"

After making this initial breakdown into classes, (ClA), (ClB), and (ClC) are normalized so that the new form of the question starts with the relative pronoun (or interrogative word), or contains it in an initial prepositional phrase. Thus:

> "If 4 cars arrive at a 4-way stop at the same instant, who has the right-of-way?"

becomes:

> "Who has the right-of-way if four cars arrive at a 4-way stop at the same instant?"

while

> "A no-passing zone is marked by what on the pavement?"

becomes:

"By what, on the pavement, is a no-passing zone marked?"

If a question is not Class 2, C1B, or C1C, then it must contain a relative pronoun. If there is only one relative pronoun, this pronoun must "ask the question," and not be in a subordinate position. When there are more than one relative pronoun, we normalize the sentence so that the initial relative pronoun "asks the question," i.e.,

> "When the yellow light in on, and you have just entered the intersection, what should you do?"

becomes:

> "What should you do, when the yellow light is on, and you have just entered the intersection?"

In this case, "when" gives information and doesn't ask for information as "what" does.

A program has been written to do this classification and normalization of questions automatically. The data consists of 195 questions directed towards the "Rules of the Road" as asked by 7 different people. All questions are classified as follows:

	Number	Percentage
Class 1		70.8
Relative Pronouns	124	63.5
What	54	27.7
Why	4	2.1
Which	10	5.1
How, How many, How much	24	12.3
When	19	9.7
Where	2	1.0
Who, For whom, Whose	11	5.7
Blanks	14	7.3
Imperatives	7	3.6
Class 2	50	29.2

At this point further analysis is applied to a subset of questions, namely, the subset of questions containing the relative pronoun "what." This subset seems attractive for two reasons. First, a relatively large percentage of the questions are of this type. Secondly, each of the relative pronouns can be expressed in terms as follows:

Original	Replacement
What	What
Why	For what reason
Which	What
How	In what manner
How many	What amount
How much	What amount
When	At what times
Where	In what places
For whom	For what person
Whose	What person's

In respect to the "Rules of the Road," these correspondences are much more specific, e.g., "where" might correspond to "on what road."

The "What" questions fall into four sub-classes from which to retrieve meaning.

1. Questions of the form: "What N V S"

Example 1A:

> "What markings are used for the center line of a four-lane highway?"

This sentence would be deciphered as:

N = "markings"

V = "are used"

S = "for the center line of a four-lane highway"

The set of all things equal (modulo V) to S would then be found, and the intersection of this set with N would give the answer. This set might include: division, double yellow line, median, etc. Since double yellow line is a marking, this would be the answer. It is not claimed that the intersection would be singular, and the number of elements in the intersection would be a measure of the ambiguity of the question.

Example 1B:

"What signs are associated with a no-passing zone?"

Interpretation:

N = "signs"
V = "are associated with"
S = "a no-passing zone."

The set might include: danger, SIGN: DO NOT PASS, slow down, yellow line next to dashed white line, etc.

Answer: SIGN: DO NOT PASS.

2. Questions of the form: "What V_1 N V_2 S"

Example 2:

"What do you do when you approach a red light?"

This question is among the last four questions that were added in anticipation of questions that could be asked, but of which we had no examples.

For interpretation:

$V = V_1 V_2$

and this sentence is of the form, "What V N S." Then the set might include: stop, decelerate, shift gears, etc.

Answer: "Stop, shift gears."

Since when NOT S occurs, you also shift gears but you don't stop, the best answer, stop, could be arrived at. This method of eliminating ambiguity probably would not always work, but it looks promising as a first approximation.

3. Questions of the form: "What be (V) S"

Example 3A:

> "What is the meaning of a small white rectangular sign marked with the letters JCT?"

This sentence would be deciphered as:

be = "is"

(V) = "meaning" (V) = ∅ means: definition

S = "of a small white rectangular sign marked with the letters JCT."

The set of all things equal (modulo V) to S would then be found. In the computer dictionary we would assume that "meaning" is the same as "equals," giving the set: crossroad, danger, etc.

Example 3B:

> "What is a motor-driven cycle?"

The set equal to "motor-driven cycle" would then consist of: "a motor vehicle with two wheels with less than 150 c.c. engine displacement."

4. Questions of the form: "What AV N (V) S"

Example 4:

> "What must a driver do before passing through a green light?"

The sentence would be deciphered as:

AV = "must"
N = "a driver"
V = "do"
S = "before passing through a green light."

Then S would imply (modulo V) a set of things, whose intersection with N would give an answer. The set might include: decelerate, slow down, get into the proper lane, etc.

In this case the answer would be: "slow down, get into the proper lane."

In the future, we plan to look at what we call "S." Current thoughts are towards feeding "S" into a parser, or looking at keywords contained in "S." Since how we get meaning out of "S" appears to be closely related to how the "Rules of the Road" is structured into memory, we plan to look at this also.

(iii) Syntactic Analysis

K. Biss, J. Schultz, R. T. Chien

This work is a continued effort to find fast and efficient methods of assigning a structure to natural English sentences, especially those sentences found in the Illinois Driver's Manual--"Rules of the Road," so that when stored in a computer, the machine will be able to answer questions on "Rules of the Road."

In the September 1967 to February 1968 Progress Report, we outlined a method for normalizing natural English text. Let us briefly review the method.

The set of sentences of a document are divided into two categories, predications and non-predications. Predications are sentences of the form "A is in some way a function of B" and non-predications are sentences which do not have this structure. A particular function in a predication can be expressed in many ways. In order to normalize the text we replace words that express the function by the function itself. Thus in the sentence "A car is an automobile" the word "is" expresses the function equality. The normalized sentence then becomes "A car equals an automobile."

We tried to apply the above method to the sentences of "Rules of the Road," hoping that those sentences which were non-predications would be easy to normalize. However, we found that only a few sentences from "Rules of the Road" could be normalized in this way and that the normalization was very involved due to the nature of the sentences. For these reasons it was not considered worthwhile to attempt to program this method of sentence normalization and we began looking at dependency analysis and immediate constituent analysis.

The basis of immediate constituent analysis is that adjacent substrings of a sentence are syntactically related. The relation is defined by the rules of the immediate constituent grammar used in the analysis. These rules are of the form $A_j + N \rightarrow NP'$ which indicates that an adjective and a noun appearing next to each other in a sentence can be combined to form an NP'. For example, let us analyze the sentence "The old man sat on the bench" assuming we have the rules:

1. V + PP → VP
2. P + NP' → PP
3. A_j + N → NP'
4. T + NP' → NP
5. NP + VP → S

Putting parentheses around the combined terms, we would get the following parsing:

(The (old man)) (sat (on (the bench))).

This analysis can be represented by the tree of Figure 1, although we then lose the information about the order in which the rules are applied.

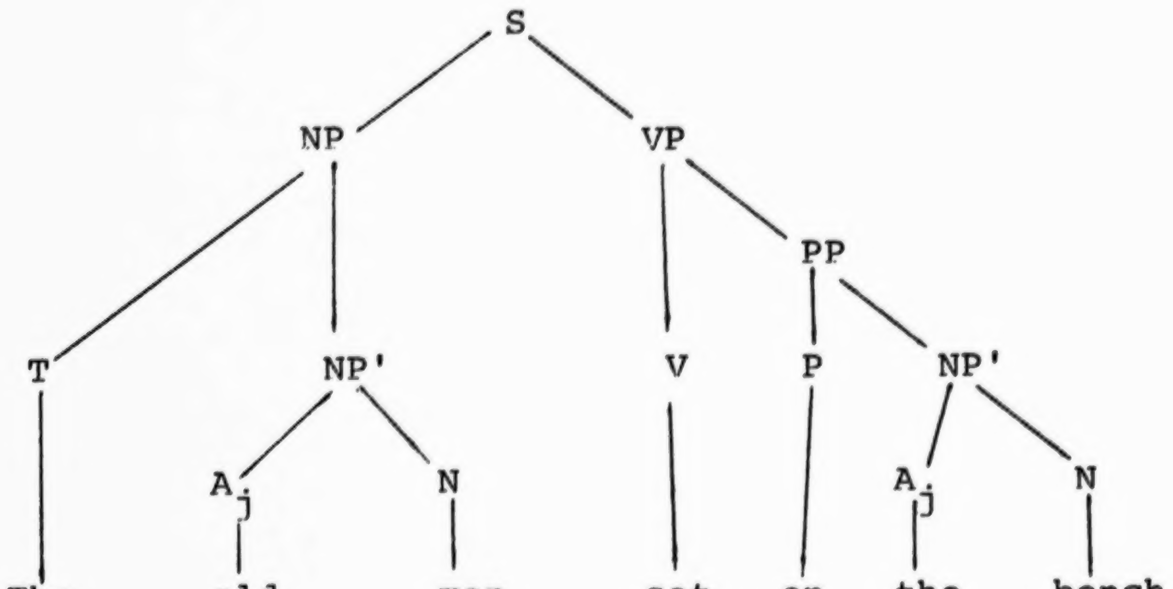


Figure 1

In our immediate constituent analysis program, our method is quite similar to that of John Cocke(1) of IBM Research. The basic plan of this approach is to build all possible two-occurrence phrases, then all possible three-occurrence phrases, etc. The syntactical structures of an n-occurrence system are the k-occurrence phrases.

The input to the program is the individual words of a sentence. First, all the words of the input sentence are looked up in a dictionary to assign the words all possible parts of speech. Next, all consecutive pairs of 1-occurrences of words are examined with all possible grammatical interpretations. For example, let us parse the sentence "The old man sat on the bench."

Form	Grammar Code Symbol
The	T
old	A
man	N
sat	V
on	PP
the	T
bench	N

In this example the first two 1-occurrences, T-A, are examined and are tested for agreement. The test is made by table look-up. The table that Cocke uses is a list of ordered triples of the form A + N → NP'. If the pair shows agreement, it becomes a 2-occurrence phrase and is stored. If not, the second word is checked with the third A-N. After all 2-occurrence phrases are stored, the program is checked for 3-occurrence phrases. The program performs this by checking the 2-occurrence phrases with the consecutive 1-occurrence phrases and vice versa. The program continues on the length of the phrases to be constructed until phrases as long as the sentence have been constructed. At each occurrence length, say X, it looks at all occurrence phrases of length n, n=1, 2,...,X-1, immediately followed by X-n. For our example the output of this program would be:

Item	Starting	Location from	Length	Grammar Code	Constituent Items
1	1	The	1	T	
2	2	old	1	A	
3	3	man	1	N	
4	4	sat	1	V	
5	5	on	1	P	
6	6	the	1	N	
7	7	bench	1	N	
8	2	old man	2	NP'	2,3
9	6	the bench	2	NP'	6,7
10	5	on the bench	3	PP'	5,9
11	1	the old man	3	NP'	1,8
12	4	sat on the bench	4	VP	4,10
13	1	the...bench	7	S	11,12

The main differences between our program and Cocke's are: a) our program is written in CSL6 language while Cocke's is written in Fortran, and b) our program will give only 1 parsing. We eliminate the possibility of several parsings by allowing each constituent to be a member of only one larger constituent. That is, once a particular constituent is integrated into a larger constituent, it is no longer a candidate to appear in any other constituent.

The reason why we chose Cocke's program is because it is so much faster than any other immediate constituent analyzer. We felt that we can even improve upon the time by not caring about ambiguities.

A dependency analysis consists of finding a tree which represents the structure of the sentence being parsed. The main verb of the sentence is taken as the head node of the tree with other words in the sentence appearing at the other nodes of the tree. The connections between the nodes indicate dependency where word A depends on word B, or B governs A, if A modifies or complements the meaning of B. For example, the sentence "The old man sat on the bench" would be represented by the following tree:

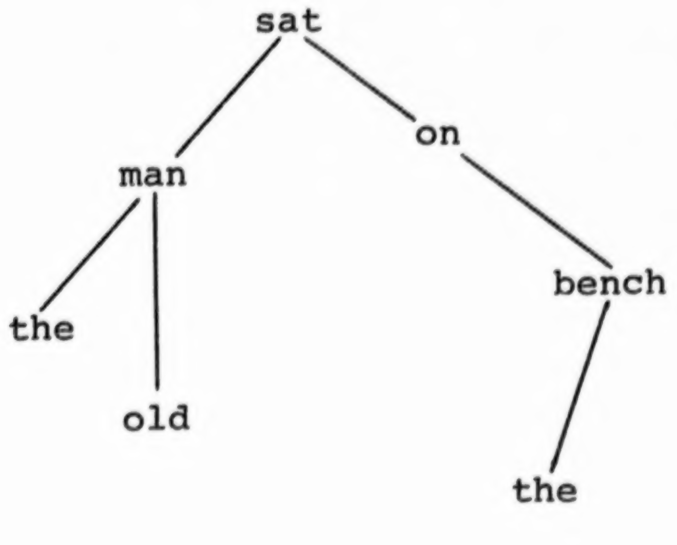


Figure 2

In this example "old" and "the" depend on "man," while "man" depends on "sat," etc.

Let us look more closely at the tree of Figure 2. If we drop a projection line from each node to a base line, we see that no projection line crosses a dependency link (see Figure 3).

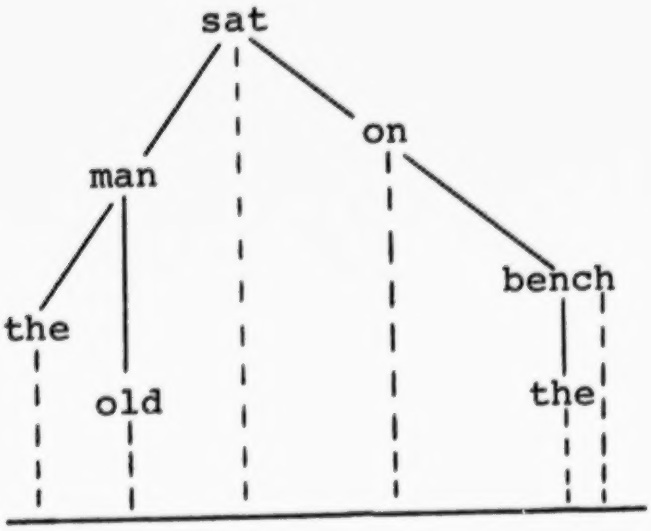


Figure 3

This property is not a universal property of English sentences, however, the assumption that all sentences satisfy the property of projectivity is like the assumption that a sentence is made up of immediate constituents.

Hays(2) has written a parsing program using a dependency grammar. The program works on the premise that two words will not be connected in a dependency tree if this connection will lead to a non-projective tree.

The first step in the program is to find a grammar code symbol for all words in the input sentence by dictionary look-up. Then using the string of grammar code symbols the program tests all pairs of symbols for precedence where precedence is defined in the following way:

X precedes Y if and only if

1. X is to the left of Y in the sentence
2. Any words between X and Y depend on either X or Y (This insures projectivity.)
3. Neither X depends on Y or Y depends on X (This insures that there will be no circular connections.)
4. Either X or Y or both are independent.

The program then makes the following computation on all pairs of words to find if X depends on Y.

X depends on Y if and only if

1. X is independent
2. X precedes Y or Y precedes X
3. There is an entry in the dependency table which states that the grammar code symbol of X can depend on the grammar code symbol.

When a dependency connection is made, the grammar code symbols for both X and Y are changed to show the properties of the pair. In this way Hays was able to obtain an immediate constituent analysis along with the dependency analysis. The final output of the program is all possible dependency parsings for the input sentence along with the associated immediate constituent analysis.

Simmons(3) has also written a program called the Pattern-Learning-Parser (PLP) which uses a dependency grammar. All of the grammar rules which this program uses are learned from a hand-parsed text.

To be more specific,the words in the hand-parsed sentences are each given a number. This number indicates the level at which the word occurs in the dependency tree of the hand-parsed sentence. For example, the words of the sentence
"The old man sat on the bench" would be assigned the numbers
-3 -3 -2 +1 +2 -4 +3
which appear below the words (note that the words on the object side of the verb are positive). The program then generates, for each word in the hand-parsed text, a word class of the form a/b/c/ and a set of sentence-pattern rules which indicate the structure of the dependency tree of the hand-parsed sentence. The word class a/b/c that is associated with a particular word indicates that the word occurred at level a in a tree, that the word appearing ahead of it was at level b, and that the word that followed it appeared at level c.

When a sentence is given to the computer to parse, the word classes for each word in the input string are looked up. By successively applying logical addition and logical subtraction to the word classes a prediction is made about the depth at which each word of the input string might appear. If there is an ambiguity, the sentence-pattern rules are applied.

This program gives multiple parsings for an input sentence, but Simmons only uses the first possible analysis.

It has been proven by Gaifman(4) that dependency grammars and immediate constituent grammars are weakly equivalent. That is, they generate the same set of sentences from the same initial vocabulary or, analytically, they classify the same string of sentences. However, it can be shown that dependency grammars and immediate grammars are not strongly equivalent. That is, there does not exist an isomorphism between the structural diagram each associates with a given sentence. Since an isomorphism does not exist, a sentence which may be ambiguous (have several structural diagrams associated with it) in an immediate analysis may have but one dependency tree, and vice versa. Therefore, dependency analysis and immediate constituent analysis each capture something of grammar which the other misses. It is for this reason Hays chose to use both dependency analysis and immediate constituent analysis.

Part of our project is to find out if it is really necessary to use both types of analysis. We are not completely convinced that, in the R2 system, two types of structural analyses will give much more information than one type of analysis. It is to test this premise that we are developing both an immediate constituent parsing algorithm and a dependency-parsing algorithm.

The dependency-parsing program is running at this time. Our program is fast and discovers, for any input sentence, the most probable parsing. The program works in the following way.

The words of the input sentence are looked up in a dictionary to discover their syntactic word classes. Using the string of syntactic word classes, we search for the governor of a particular word, checking the words first at distance one from the given word, then at distance two, etc. A word and its candidate for governor are checked in a table of possible dependency connections to find if the syntactic class of the candidate can govern the syntactic class of the particular word whose governor we are trying to find. Once a governor is found for a word we make the connection between dependent and governor and go to the next word.

We have chosen this way to parse over the approaches of Hays and Simmons for several reasons. Both Hays' program and Simmons' program gives multiple parsings. Simmons arbitrarily picks the first parsing as the correct one. By searching for the most probable governor, we eliminate questions of ambiguity but still get a "good" parsing quickly.

At this time our program can deal only with simple sentences. However, we are working on subroutines to deal with compound and complex sentences.

(iv) Context Modeling

R. T. Chien, D. Lombardi, F. P. Preparata

In trying to realize a model that could be used to logically structure the "Rules of the Road" in a data base, it was first necessary to make some preliminary judgments concerning the major divisions to be handled. It was found the entire text could be divided into 5 categories that would span all conceivable topics. These 5 categories are listed below:

1. legal requirements
2. motor vehicles
3. traffic laws
4. accidents
5. unsafe driving practices and emergency action

These categories are obviously not mutually exclusive, but each has distinct constituents that make it a definite advantage to try to deal with them separately.

In an attempt to derive a successful model for arranging information concerning traffic laws, to begin with, it was necessary to make a further classification. Traffic laws can be subdivided into the following topics that are necessary (although possibly not sufficient) for an adequate coverage.

1. right of way
2. passing
3. lane usage
4. turns (& signals)
5. speed restrictions
6. stopping
7. parking
8. trucks, buses, trailers, taxis, etc.

A further inspection of these subtopics reveals that each can be broken down into two basic components concerning traffic law: 1) an explicit component, that can be uniquely defined by a highway sign, a traffic signal and/or a pavement marking; and 2) an implicit component that is known _a posteriori_. For example, there are specific highway signs and signals that regulate turning and indicate exactly where and when it is legal to make a turn. However, to turn safely and obey the traffic laws, it is also necessary to make up your mind before you get to the turning point, move into the proper lane as soon as possible and to slow down to a "reasonable" turning speed. These last three prerequisites for a safe and legal turn comprise the implicit component of the subcategory "turns (& signals)." Therefore, it is apparent that if some systematic structure for modeling traffic laws is to be devised, it must be capable of handling both of these components successfully.

Some work has already been done on modeling the explicit component. This was a relatively simple task since in this category we have a physical basis for establishing the various relationships between the entities involved. As an example, a model for arranging information concerning highway signs was devised and is described below.

It was first necessary to develop an attribute-value property list for each sign. It was found that each sign analyzed could be completely described using five-attribute value pairs. These pairs are listed below for a STOP sign.

Stop

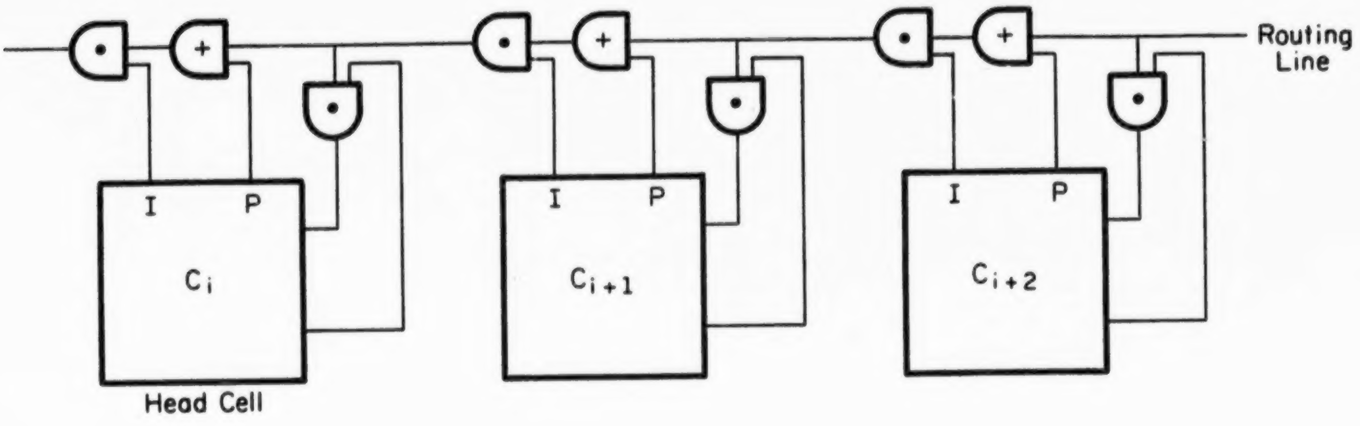
(shape: octagon)
(color: red with white letters or yellow with black letters)
(location: intersections)

(purpose: regulate traffic)
(drivers reaction: to bring vehicle to a complete stop before entering a marked crosswalk or crossing a marked stop line. After stopping he must not start again until he has yielded the right of way to pedestrians and to approaching traffic on the through highway, and until a safe interval occurs)

Any information concerning an attribute of a particular sign can be modeled using an attribute-value property list similar to the one shown above. For example, the above list contains all the necessary information for answering a question of the type: "What is the shape of a STOP sign?". However, other types of questions concerning signs must also be evaluated. If the question does not refer to a specific sign, but is in regard to the general nature of signs, the model must be capable of keying on certain words and making the appropriate information available. This means that it is necessary to interrelate all the information in such a way as to make all the necessary information available regardless of the point where the data array is entered.

The following diagram Figure 4 illustrates how this information can be interrelated.

It is easy to see how a model of this type could be extended to include traffic signals, pavement markings and, in fact, any physical aspect of driving that can be described by attribute-value pairs. However, when it is attempted to extend this model to include the implicit component, it fails. At present, a scheme based on cause-effect relationships is



Routing Line
I
P
C_i
Head Cell
I
P
C_{i+1}
I
P
C_{i+2}

Figure 4. A Typical Organization

being tried to see if it would permit a systematic structuring of the implicit component. Once a successful scheme is devised, it will be necessary to try to effect a compatible intersection between the different models and thereby lead to a successful overall model.

(v) Concept Processing

F. Stahl, J. Jansen, R. T. Chien

Initial processing has been completed on the data base for "Rules of the Road" with respect to the maximal phrase relational strategy. The objective is to find a general strategy technique with a high relevancy rate for retrieving statements from the data base, given a question that can be answered by using the statements from the data base.

A strategic relevancy rate has been defined as N/(A+E); where A is the number of statements in the data base that are relevant; N is the number of relevant statements that the given strategy technique actually encountered, note $N \leq A$; E is the number of statements that the given strategy encountered, but that are not relevant.

The maximal phrase relational strategy consists of a dictionary, a set of statements, and a strategy.

Each entry in the dictionary contains a maximal phrase and a set of pointers that refer to statements.

Maximal phrases are arrived at in the following manner:

(a) Sentences of the data base are numbered consecutively.

(b) A WIS Index is processed on the sentences from a.

WIS means words in sentence, its name is derived from KWIC where we consider all words not just key words and the context is a sentence.

(c) Consecutive entries of the Index are compared to find the longest string of words that match. Thus, if

STOP AT A STOP SIGN...239 and

STOP AT A RED LIGHT...646 were consecutive entries, then

STOP AT A 239, 646 would be recorded.

(d) Remove the prefixes from the output of C. If one entry is the beginning part of another entry, and if these two entries have any numbers in common, then the numbers that are common to both are removed from the former. If an entry results with no numbers, it is deleted, that is:

Stop	239, 362, 424, 749
Stop at	239, 646
Stop at a	239, 646, 932
Stand in	136, 426

would result as

Stop	362, 424, 749
Stop at a	932

(e) The output from d is reverse sorted. That is, if an entry were

Stop at a

it would be sorted as if it were spelled

a ta pots.

This results in a list of entries that have similar endings like:

go at a

stop at a

halt at a.

(f) The suffixes from the output of e are removed. This is an analogous operation to removing the prefixes in d.

Thus if the book	269, 348
and of the book	269, 348, 729

were entries, then

of the book	269, 348, 729

would be recorded.

(g) The output of f is resorted in normal alphabetical order. There are a number of strategies that are being tested for relevancy rate. Given a question:

- (a) Find the maximal phrases of the question and retrieve the indicated statement.
- (b) Find the maximal phrases of the question and retrieve all those statements that occur with at least two pointers, three pointers, etc.
- (c) Retrieve statements by a, or b, and consider the maximal phrases of these statements and retrieve those indicated statements.
- (d) Same as c, only retrieve statements, with technique of double occurrence of b.
- (e) Find the maximal phrases of the question and for each pointer n also consider pointers n-1 and

n+1 (the statements have same clustering!).

(f) Same as e, using double occurrence of pointers as retrieval criteria.

2. Grammars and Relational Structures - P. Weston

Most of the work has been completed on the basic interactive program, named "LEJARDIN", for graphic manipulation on the CRT console. Using both typewriter and light pen control, LEJARDIN allows the definition of arbitrary designs composed of line and point elements and their assembly into logically composite structures. In addition, complex arrays of graphic elements can be formed and rearranged through nothing more than variation of numerical parameters in an otherwise fixed composite structure. This feature will be beneficial not only in immediate drafting applications but also in the longer term application of supplying an 'environment' for an interactive language acquisition program which can be taught a form of natural language in interaction with a human operator. When time permits a manual for LEJARDIN will be prepared.

A manual has been prepared for CYL6, the CYLINDER data structure system described in the previous quarterly report. This manual is in press at CSL and will be available early in November.

In addition to the above projects, the design of a linguistically interactive program called SMALTALK has begun.

This is intended to supply the second major component in the final design of the projected language acquisition program, the first component being the now largely complete graphic program, LEJARDIN. SMALTALK will serve as a vehicle in developing that notion of grammar which is implicit in the proposal for this project. The foremost principle in this concept of grammar is that the formation rules of linguistic utterances are best seen not as constraints within the domain of message elements, no matter how abstract or indirectly inferred these elements may be as in the grammatical theories following Chomsky, but rather as the result of what we may call expression rules which carry relational structures representing message content in the mind of the speaker to their conventional message structure representations, i.e., the speaker's expression of this content. This view eliminates two familiar features of current grammatical theory; the distinction between surface-structure and deep structure, and the notion of grammatical transformation. It also reduces the importance of the sentence as a fundamental message unit.

The reason for elimination of the deep structure concept as a syntactical notion is that it appears possible to assign a relational structure significance to all surface elements, so that the theory may be based upon operations which act directly upon the relational content structure, eliminating the original theoretical need for syntactical intermediaries to support separate 'semantic projection rules.' This implies among other things that a relational meaning must be found for the so-called meaningless words, e.g., 'the', 'of', 'is', since they are prominent and ubiquitous surface elements. This does appear possible, and in addition modal auxiliaries and verb conjugations are similarly accomodated.

In the attempt to discover the form of relational structure needed to support such a theoretical treatment it has been found necessary to take into account information bearing upon not only the message itself and its explicit content but, in addition, the framework within which the message is perceived by the hearer. Pertinent to the latter category are included: (a) the credibility of the source, i.e., whether fictional, deceitful, or simply factual (when the modal content of the material makes this distinction relevant); (b) the directness or indirectness of the source, e.g., written material, direct communication, indirect communication through a third person, etc.; (c) the modal content of the message, in which the speaker's attitude toward the content may be expressed, including certainty, possibility, desirability, promised action, question, and so on. In most of these cases truth or falsity is actually an irrelevant consideration; (d) connections between separate statements including logical ones expressed by the familiar connectives 'and', 'or', 'if... then...', etc., as well as those coupled with modal information such as 'although.' Furthermore the contextual connections between statements on two sides of a dialog, often transmitted through ellipses rather than through marked grammatical elements, is relevant in this regard; (e) indicators relating the message content to the conventional metaphysical framework of the culture whose language is being spoken. This includes time reference through verb tense, aspect, and phase, as well as indications of space reference and other aspects of action such as agentive or ergative role, all of which have been indicated by case systems in the grammars of various languages.

All of the above considerations (a) through (e) are largely independent of the ordinary semantic problem of the reference of so-called content words, the open-class vocabulary, which also must be handled in some manner to achieve a working simulation. In light of the distinction between content and framework and its reflection in the distinction between closed-class and open-class lexical entities, the term 'canonical' has been adopted in referring to the message framework, logical, modal, metaphysical, which is considered by the native speaker as _a priori_ in character. This term is opposed to the term 'empirical' referring to the meaning of content-words, involving much information of an empirical nature.

It is a working assumption that the _a priori_ framework, or canonical structure which we have postulated, can be formulated in a sufficiently language-independent manner as to serve in the explanation of language universals. This assumption is plausible because the relevant cognitive activity lies not entirely within the linguistic domain but rather at the boundary between linguistic and extra-linguistic behavior and therefore not entirely language determined. It is also a necessary working assumption in proceeding toward the earlier-stated goal of a linguistically adaptive machine, which, to be non-trivial,requires a language-independent formulation.

3. Investigation of Fundamentals of Nonlinguistic Cognition - F. P. Preparata, S. R. Ray

In this section, we describe an attempt to distill the mass of requirements for a cognitive system down to certain essential problems. Of first importance is the tentative conclusion that the representation of meaning of a communication must be based upon knowledge of the natural constraints between objects and relations, that is, upon a world model or the "eidology", according to the terminology introduced in "Cognitive Memory: An Epistemological Approach to Information Storage and Retrieval" (project proposal).

This is the direction which Chomsky takes in "Aspects of the Theory of Syntax" by supplying property markers. We feel, however, that Chomskian markers are far too inflexible as a mechanism for representing a world model. Rather, it appears that an approach guided by the philosophy of cognition expounded by Lenneberg in "Biological Foundations of Language" is much better suited to our task.

The principal message of Lenneberg, as it applies to this investigation, is that cognition is based upon a nonlinguistic discrimination process with verbalization as a largely extraneous (or, certainly, secondary) appendage.

In the same frame of concepts, a widely-accepted model for the description of cognitive processes contains two main transformational components, that is: 1) a mapping from the environment through percepts to the eidology; 2) a mapping from the eidology to the linguistic description, as well as its inverse. It is apparent that the two mentioned mappings "environment → eidology" and "eidology → language" have distinct characteristics which warrant a separate analysis. Furthermore we feel that the insight gained through the analysis of the perceptual mapping may greatly facilitate, in due time, the study of the linguistic mapping.

We were accordingly led to attempt to synthesize and study a very simplified model of a concept-discrimination process. The percepts chosen as stimuli (input) are stylized versions of "scenes" of an anthropomorphic environment or "real world" containing representations of shapes, colors, sizes, positions, etc. (i.e., visual scenes transformed so that these properties need not be graphically discriminated). The "real world" assumed as the underlying matrix of the percepts bears no character of necessity, but is only chosen for its convenience in view of an empirical test of the results of the investigation.

The final objective of this study is the development of a procedure for cognition, namely an analysis mechanism which converts a given scene into a description thereof, this description being partially classificatory and partially relational. It must be noted, however, that the analysis problem cannot be tackled prior to the precise definition of the "universe" upon which the cognitive mechanism is to operate. This clearly leads to the synthesis problem, namely to the description of the model which distinguishes admissible from inadmissible scenes. We can therefore distinguish a "generator model" and a "cognitor model" (usually contained within the former), which, roughly, play the roles of speaker and hearer, respectively, according to the well-known linguistic terminology.

The crux of the problem is therefore reduced to the specification of the nature of the "generator model" and of the embedded generative device. Intuition would suggest that the grammatical constraints should be rather weak and that the interdependence of the "details" should be rather strong, which is equivalent to strengthening the semantic constraints. How to express the weakening of the grammatical rules, or even more, the strengthening of the semantic texture is presently unclear. It appears practical and intuitively appealing, however, to express the constraints on the choice of details (i.e., the constraints on the rewriting rules) as determined by some broad properties of the remainder of the scene, instead of by the specific details occurring elsewhere in the scene itself.

The tentatively outlined structure is reflected, although in a somewhat incomplete form (which leaves the door open to some learning capability), in the cognitor model. Upon presentation of scenes (percepts), it is necessary to disambiguate the communication by successively extracting properties, and attempting to infer the description of finer details on the basis of accumulated inexact data from the "world-model." The disambiguation will proceed from the general toward specific although backtracking (changing a guess) is a necessary provision, under the guide of a currently computed likelihood function (the strategy being very much analogous to the one used in sequential decoding). For example, the time and general location of a scene will first be guessed; these choices help to restrict the probable kinds of objects and their positions, etc.

At the present stage of the work, we wish to answer:

1) Which is an adequate structure for the implementation of the generator model (and, consequently, of the cognitor model)?
2) How to proceed to construct a description of a particular event (communication) as it is disambiguated?

4. Cognition and Heuristics - H. Bielowski

The formation and use of concepts is assumed to be the main feature of cognitive systems. Research has been made towards mechanisms capable of acquiring concepts. As a working hypothesis, the following definition by Church (1956) has been adopted: a concept is a decision rule to be used to decide whether or not a given object is a member of a certain set of objects to which the name of the concept applies.

It is assumed that the acquisition of concepts must be motivated and controlled by the criterion of usefulness.

As a method of study, it has been decided to work on the outline of a tentative computer program for the solution of a specific class of problems within a simple universe. The class of problems chosen is the one of assigning two-dimensional objects to locations on a plane with the goal of minimum connection cost, if these objects have to be connected in a given manner, and the connections cost is proportional to the distance to be bridged. The layout of etched wiring boards or the placement of machines in a factory are practical examples of this problem.

There does not exist a solution algorithm for these problems, except, in some cases, complete enumeration of the solution space, which is unfeasible for more than, say, ten objects.

Restricted permutations are the most effective methods today. Unfortunately, they are qualitatively and quantitatively restricted to only the simplest assignment problems.

The point in these problems is to achieve a match between the topological structure of the connected objects and the geometrical pattern of the available spaces. Clearly, this involves the ability to recognize topological structures as well as geometrical shapes.

Work is now in progress on the definition of principles and procedures which would enable a system to learn and apply simple topological and geometrical concepts, as "chain, tree, ring," etc., and "corner, square, channel," etc.

As a sideline, a 1000-statement FORTRAN program named COBSOP (Connectivity Based Sequence of Placement) has been written. It serves for the study in more detail the handling of topological data and the degree of recognition obtainable with fixed, built-in routines. In addition, the program could be used as a tool for solving actual assignment problems.

5. Machine Architecture for Information Retrieval - A. Kisylia, R. T. Chien

(i) Introduction

The object of this report is to describe an information-retrieval system which operates in a highly parallel fashion in an associative mode. By associative it is meant that a particular item is located in the data base by content addressing. In this process, an entire record of information is located using a known part of the record as the search criteria. As an elementary example, the record "John's car is bright blue" could be located or retrieved by specifying the words "blue" and "car." This method of data interrogation is desirable in all information-retrieval problems. The notion of addressing by contents enables us to avoid the artificialities of location assignment and frees us to a large extent from such local considerations as scanning, searching and counting; the operation is parallel due to the fact that the entire content of the memory is searched simultaneously. This is accomplished by distributing a sufficient amount of logic throughout the memory. This allows each storage location or "cell" to act, to a certain degree, as a small, independent, processing unit.

The associative memory processor described below is a logical outgrowth of the organization proposed by Lee and Paull(14). This previous organization was thoroughly investigated and it was found somewhat deficient in two areas. First of all, the general operation of the processor was found to be inefficient in terms of the number of steps (therefore the amount of time) and the amount of programming required to perform a data search. Secondly, but most important, it was found that certain types of searches which are extremely important to information retrieval could not be performed at all. The next section briefly describes the organization and operation of the processor presented by Lee and Paull. The shortcomings of this system and qualities which are desirable in a processor of this type are also presented. The final section describes in brief an associative memory processor which alleviates the problems presented and therefore facilitates a more efficient information retrieval system. Examples are given throughout to clearly explain the operation.

(ii) Investigation of the Processor of Lee and Paull

The machine proposed by Lee and Paull consists of an associative memory connected to a conventional, stored-program computer. The primary function of the computer is to act as a control device for the associative memory. It also must be able to store and execute the search programs and temporarily store data which is retrieved from the memory. The data base which is to be interrogated is stored in the associative memory.

The basic memory is a linear array of small, identical, sequential machines called cells. Each cell contains both a number of binary storage elements and a sufficient amount of logic to enable it to perform the functions of the cell. The storage elements of the cells are of two types: the symbol elements and the activity elements. The symbol elements are those in which the information or "symbol" of the cell are stored. The activity elements are used as bookkeeping tags to keep track of particular cells during the operation of the machine. Each cell in the array is connected to a set of common bus lines, which include the input leads, the output leads and the various control leads which provide the cells with commands. A cell C_i also has the ability to transfer its activity status to either of its immediate neighbors, C_{i+1} or C_{i-1}. The contents of every cell (the symbol and/or the activity) can be matched against the pattern presented by the control computer on the input bus lines. In this way all cells containing a particular bit pattern can be located and tagged.

Information is stored in the computer as strings of symbols with each symbol stored in a different cell. (Strings can be of any length as long as there is room to store them.) Consecutive symbols of a string are stored in consecutive cells. Therefore if a string contains n symbols and the first symbol of the string is stored in cell C_i, then the second symbol is stored in C_{i+1}, the third in C_{i+2}, and so on with the nth symbol being stored in the C_{i+n} cell. The location of a string is completely arbitrary since the cells do not have addresses, but the order of the cells in which consecutive symbols are sotred is very important. A special symbol α_0 is provided for the beginning of a string. Parts of strings or parameters are distinguished by storing another special symbol, β, after them. Therefore if a string consisted of the parameters XY, OW, PHD, it would be stored in the cell memory in the following manner: $\alpha_0 XY\beta OW\beta PHD\beta$, with α_0 stored in cell C_i, X in C_{i+1}, Y in C_{i+2} , and so on.

Identification and retrieval of a string or a particular set of strings are accomplished by using a particular parameter as the search criteria. For instance, if it were desired to locate all strings which contained the parameter OW the search would proceed in the following way. First, every cell in the memory would match its contents against the input bus lines which would contain the pattern β. This denotes the beginning of the parameter $\beta OW\beta$. If a cell has a successful match it will set an activity bit. At this point all cells containing β

have an activity bit set. Next, all cells with activity bits set are told to propagate this activity to their neighbor with a higher index (to the right) and then reset their own activity. Now all cells compare their contents with the input pattern consisting of a symbol equal to 0 and a set activity bit. Only those cells which contain a set activity and the symbol 0 are allowed to keep their activity set. All other previously-active cells are reset. At this point only those cells which are at the beginning of parameters and contain the symbol 0 are active. Again the activities are propagated to the right, and search is made for all active cells containing the symbol W, with all other activities being reset. Now only those cells containing W which is the second symbol of all parameters having 0 as the first symbol are active. Finally, the activity is again propagated to the right and a search is made for all active cells containing the symbol β. At this point only those strings which contain the parameter OW have an active cell. This cell is the cell containing β which follows OW.

This searching process is a very efficient one. It can be thought of as eliminating useless information rather than a searching process for useful information. Although the search is done serially by character, it is a parallel operation conducted in each of the strings simultaneously.

Although at this stage all the strings containing the parameter OW have been isolated, more processing is required before these strings can be retrieved. First, a priority system must determine which of the strings of the set is to be retrieved first. After this string is found, the activity present in this string must be transferred to the cell containing the α_0, head cell, of that string. This is accomplished by propagating the activity to the neighboring cell with lower index (the cell to the left) and matching for a cell which is active and contains the symbol α_0. If the match is successful, the head cell of that string has been found. If it is unsuccessful, the process is repeated until the head cell is found. From this point, the characters in the string are read out sequentially, one at a time, by successively propagating the activity to the right neighbor and ordering the active cell to place its contents on the output lines. This process, of course, is non-destructive.

This example introduces two of the shortcomings of this processor. First, the priority system of this machine only allows one string of the set to retain an active cell. This string is the one to be retrieved. Since they no longer contain an active cell the remaining strings in the set must again be identified by another search in order to be again eligible for retrieval. Provision is made to eliminate previously retrieved strings from subsequent searches, but still the same search operation must be performed as many times as there are strings in the set to be retrieved. These repetitive searches require additional operating time during which no really useful processing is being accomplished. It would be desirable to be able to retain the active status of all the strings of a set during the retrieval of the entire set. This would require that only one search, the initial search, be made. The second deficiency to be noted here is the waste of processing time needed to find the head cell in a string chosen for retrieval. The activity must be propagated to the left and a match be performed as many times as the number of cells between the head cell of the string and the active cell which resulted from the search. If this number is large, the time involved to complete this process becomes very costly, because the rest of the memory must stand idle until its completion. This suggests that it would be advantageous if all the cells in a string could communicate directly with the head cell without the necessity of involving other cells.

Additional areas in which this processor and all other similar associative memory processors (15, 16, 17) designed for information retrieval are deficient can be placed in two general categories. The first are the operations the machine can execute but does not do so efficiently. The two mentioned above fall in this class.

Another problem would be that of identifying the set of all strings which contain two or more particular parameters. If the parameters are stored in a particular order, and they are stored in contiguous groups of cells, the searching process is the same as above, e.g., A and C are the parameters which the strings must contain, and they are stored in all strings of the set containing A and C in the form βAβCβ. If one or both of these requirements is not met (as is usually the case) then the searching process becomes quite complicated and requires a great amount of processing time.

Still another problem which has not been handled efficiently is that of repacking the strings in the memory after information has been eliminated. This is necessary so that all the empty cells in the memory can be utilized to store new strings of arbitrary lengths. The methods of gap closing proposed by the various authors involve programs which are extremely inefficient with regard to the amount of processor time it takes to repack the memory.

The second category includes those types of operations which are necessary for an efficient information retrieval system(18) but cannot be performed by the type of associative processor suggested by Lee and Paull and its descendents(16,17).

The organization described above identifies the set of all strings which contain the parameters of the search criteria exactly. It cannot perform a threshold search. A threshold search can be described as the identification of all strings which contain at least a certain number of the parameters used for the search. For example, it might be desirable to locate all strings which contain any 3 or more of 5 particular parameters.

A second type of search which would be useful would be a weighted threshold search. In this identification procedure, the presence of a particular parameter in a string may be of more or less value than that of other parameters. In this operation each parameter used in the search would be given a weight or value corresponding to its importance. If a string contains a particular parameter, this value is recorded and added to the results of previous and subsequent parameter searches. After all parameters have been searched, only those strings which contained a total search value greater than a certain amount would be of interest for retrieval. Both types of searches described above imply that the strings should be capable of storing the results of many searches. No provision is made for this in the associative memory processors mentioned above.

The following section is a brief description of an associative memory processor which will alleviate the problems outlined above and facilitate a more efficient information retrieval system.

(iii) A New Associative Memory Processor

The structure of this associative memory processor is essentially the same as the configuration proposed by Lee and Paull. The basic differences lie in the memory cell. This allows an entire string of cells to act together as an independent processing unit.

The basic configuration consists of general purpose computer acting as a control device for an associative memory. The computer stores and executes the search programs, issues commands to the control leads, and temporarily stores the data which is to be entered into or retrieved from the associative memory. The memory stores the data base which is to be interrogated.

As in the previous model, the memory consists of a linear array of small sequential state machines called cells. Each cell is connected to a common set of bus lines. These are the input leads, output leads and control leads. These lines allow the control computer to communicate with all the cells simultaneously (if the propagation time along the lines is ignored).

All cells in the memory are identically the same. Each cell contains both storage elements, binaries, and a certain amount of logic which enables it to perform the various functions required.

The storage elements are divided into four different fields. A six-bit field stores the character or symbol of the cell. This is the information register. There is a one-bit activity field. This bit is used as a bookkeeping tag to keep track of certain cells during processing. Another bit is used to indicate if a cell is empty or if its symbol contents is no longer considered important. The last bit is the α_0 bit. The presence of this bit allows a string of cells a high degree of processing power.

A cell may function in one of two states. The state of a particular cell is determined by the condition of its α_0 bit. If the α_0 bit is set the cell acts as the head cell of a string. In this state the cell is able to receive count pulses which can be emitted from all the cells of higher index up to the next head cell. The binary representation of the number of these pulses are stored in the head cell's symbol register, which now acts as an accumulator. This concept of the function of a head cell is very important. It allows the results of successive searches to be stored in each string and eliminates the necessity to propagate an activity to a cell of lower index (to the left). It also enables the memory to perform both a threshold search and a weighted threshold search with equal ease. These points will be explained clearly in the example at the end of this section.

Each cell in the memory can communicate directly with other cells in three different ways. First, a cell can transmit its set activity to its neighbor with a higher index (to the right). Secondly, as mentioned above, an active cell in a string can transmit a pulse directly to the head cell of its string. This pulse is sent along a special gated network called the routing line. This is a one directional path which passes through all the cells in the memory. Part of the routing line is shown in Figure 5. One head cell and two symbol cells are displayed. Last of all, each of the cells has the ability to shift the contents of all its registers to its

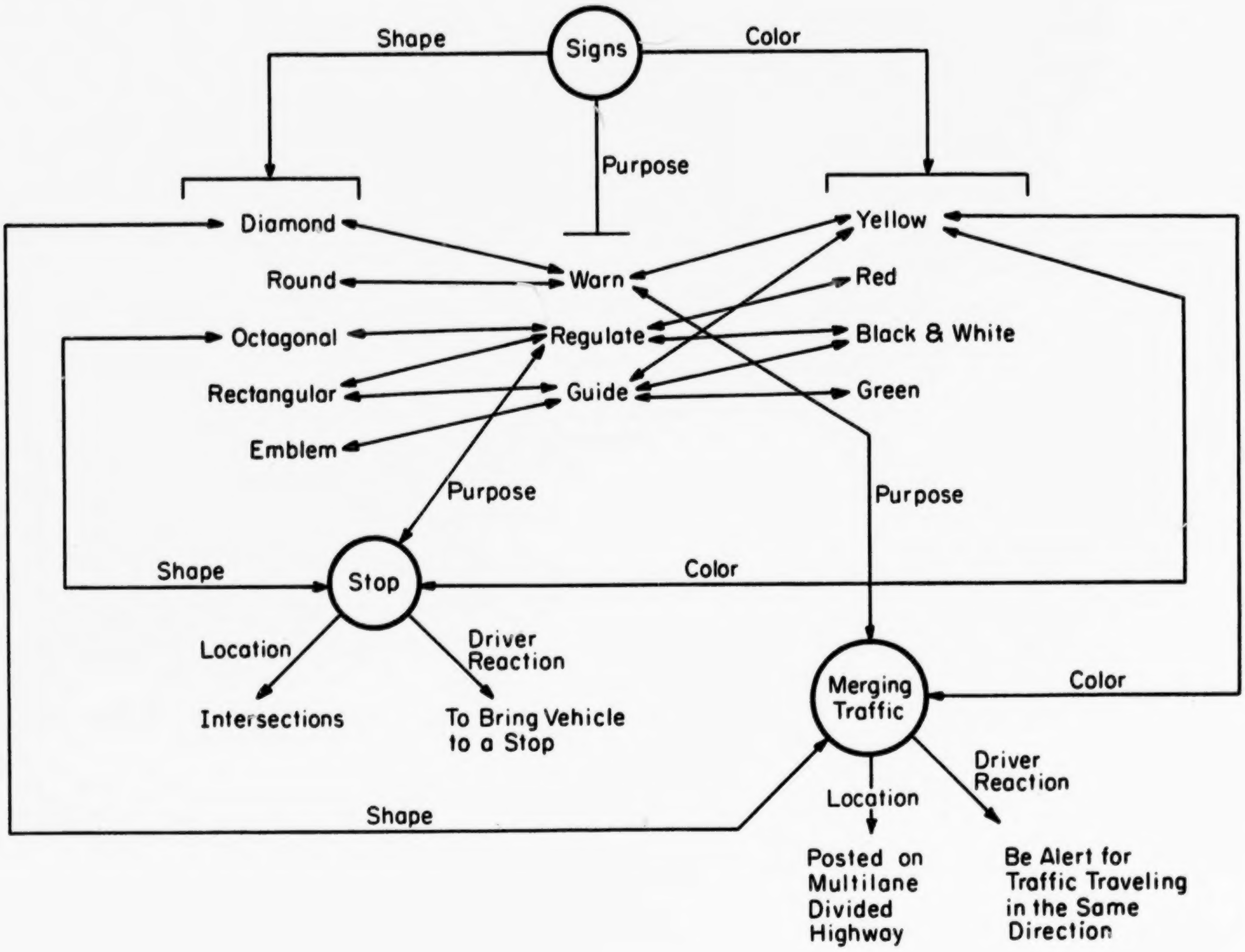

Signs
Shape
Color
Purpose
Diamond
Round
Octagonal
Rectangular
Emblem
Warn
Regulate
Guide
Yellow
Red
Black & White
Green
Purpose
Purpose
Stop
Shape
Color
Location
Driver Reaction
Intersections
To Bring Vehicle to a Stop
Merging Traffic
Color
Shape
Location
Driver Reaction
Posted on Multilane Divided Highway
Be Alert for Traffic Traveling in the Same Direction

Figure 5. A Model of the World of Signs

neighbor on the right. This is used when repacking the memory and can also be used to load or unload the entire contents of the memory automatically.

During a match, the contents of every cell is simultaneously compared with the pattern presented on the input lines. If a cell's contents matches this pattern, the activity bit of that cell is set. "Don't care" conditions can be specified in any of the bit positions so that a certain set of bits may be examined at any one time.

Information is stored in the memory as strings of characters. Each character of the string is stored in a distinct cell. The first cell in the string is always a head cell. The special symbol γ always precedes and follows the parameters of a string and the last two symbols in a string must both be γ. As an example, the string containing the parameters XY, OW, PHD, PCC, and AF would be stored in the following way:

$$---\alpha_0\gamma XY\gamma OW\gamma PHD\gamma PCC\gamma AF\gamma\gamma---.$$

As as example of operation, consider the problem of locating all strings which contain the parameters XY and AF. The process for locating a particular parameter is the same as in the example in the previous section. First, XY is located everywhere in the memory. At this point, only those strings which contain XY have a cell with its activity set. A control bus now tells all cells with their activities set to place a pulse on the routing line. These pulses are now recorded in the head cells of all the strings containing the parameter XY. Each of these head cells now contain a binary count of one in their symbol registers. The activities are reset and a search is now made to find all the occurrences of the parameter AF in the memory. After these are found only those strings which contain AF have a cell with its activity set. Again, a control bus orders all active cells to place a pulse on the routing line. At this point, only the head cells of the strings which contain both XY and AF have a binary count of 2 in their symbol registers. All activities are again reset. Now the head cells of the set of strings which contain both parameters can be activated by matching the contents of every cell with the appropriate pattern on the input lines. At this point, a priority system can be used which resets all but one active head cell. The search for the parameters XY and AF will not have to be made again, because the results of the initial search are still stored in the symbol registers of the head cells. These cells can be activated again by a simple match operation. Before a string is read out, the bits of its symbol register are reset so that it will not be considered again in the priority search.

It is evident from this example that many processing steps and therefore much time has been saved by allowing the head cells to both store the results of consecutive searches and communicate directly with the cells in their strings.

It is apparent that a threshold search would be carried out in exactly the same way. The threshold value would be used when matching for head cells which contain a particular stored number. A weighted threshold search would be performed by instructing the active cells resulting from a parameter search to issue a number of pulses equal to the relative importance of the parameter. After all searches are completed, the head cells would be tagged if they contained a value greater than a certain number.

6. Studies of the Mathematical Theory of Cognition - E. J. Scott

(i) On the Forms of Equations Associated with Inductive Inference Computers

As indicated by H. Von Foerster in various publications auch as "Memory and Inductive Inference" and "Memory without Record" (The Anatomy of Memory) to build a computer that learns from experience, it has to be "pliable" or, more accurately, it must be an adaptive computer. Whatever other aspect built into the computer it should be able to make "inductive inferences." This implies that the system should be able to compute future events from past experience. Because the next sentence as given in "Memory without Record" is pertinent and crucial to what follows, I place it in quotes. "It is clear that only a system that has memory is capable of making inductive inferences because, from a single time-slice of present events, it is impossible to infer about succeeding events unless previous states of the environment are taken into consideration." If, as is often done, we think of the "brain" as a large number of neurons which are interconnected in some way to form a complicated network, the stimulus-response equations should embody position coordinates, a time coordinate, and what is crucial, terms with time retardation which would give the computer the essential characteristic previously mentioned, namely, the ability to predict future events on the basis of a knowledge of past states. That is, to give an equation or a system of equations with initial conditions--that is, the state at this moment, say t=0, and no retarded arguments in time--is insufficient to characterize an inductive-inference computer.

It follows that systems that are capable of "predicting" or "learning from experience" ought to be characterized by equations of the form

$$\begin{aligned} \dot{x}(t) &= f(t,x(t),x(t-\tau)), \quad t > \tau \\ x(t) &= g(t), \quad 0 \leq t \leq \tau, \end{aligned} \qquad (1)$$

where $\tau > 0$ and $g(t)$ characterize the "past history." More, generally, they should be governed by a system such as

$$\begin{aligned}
\dot{x}_1(t) &= f_1[t,x_1(t),x_1(t-\tau_{11}(t)),\ldots,x_1(t-\tau_{1m}(t)),\ldots,\\
&\qquad x_n(t),x_n(t-\tau_{n1}(t)),\ldots,x_n(t-\tau_{nm}(t))],\\
\dot{x}_2(t) &= f_2[t,x_1(t),x_1(t-\tau_{1},(t)),\ldots,x_1(t-\tau_{1m}(t)),\ldots,\\
&\qquad x_n(t),x_n(t-\tau_{n1}(t)),\ldots,x_n(t-\tau_{nm}(t))],\\
&\qquad \ldots\ldots\ldots\\
\dot{x}_n(t) &= f_n[t,x_1(t),x_1(t-\tau_{11}(t)),\ldots,x_1(t-\tau_{1m}(t)),\ldots,\\
&\qquad x_n(t),x_n(t-\tau_{n1}(t)),\ldots,x_n(t-\tau_{nm}(t))], \qquad (2)
\end{aligned}$$

or, more shortly,

$$\begin{aligned}
\dot{x}_i(t) &= f_i[t,x_1(t),x_1(t-\tau_{11}(t)),\ldots,x_1(t-\tau_{1m}(t)),\ldots,\\
&\qquad x_n(t),x_n(t-\tau_{n1}(t)),\ldots,x_n(t-\tau_{nm}(t))], \qquad (3)
\end{aligned}$$

$$i = 1,2,\ldots,n, \qquad t_0 \leq t \leq B, \qquad \tau_{ij}(t) \geq 0$$

and the initial conditions (constituting past history)

$$x_i = x_{i0}(t), \qquad (4)$$

defined on the set E_{t0} consisting of the point t_0 and all those differences $t-\tau_{ij}(t)$ $(i=1,2,\ldots,n;\ j=1,2,\ldots,m)$ which are less than t_0 for $t_0 \leq t \leq B$.

The mathematics dealing with such systems has, to a certain extent, been developed. It is far from complete, however. Nevertheless, certain steps in the direction of an inductive-inference computer can be taken by considering certain functions f_i and studying their solutions. The forms of the function f in (1) and f_i in (2) would play a crucial role. What their structure should be is not too apparent at the present.

An examination of the literature indicates that my surmise about the form of the equation (1) that go with a possible prediction theory is substantiated in at least one case by an article by Grossberg(19). In this paper, the author considers a system of the form

$$\dot{X}(t) = AF(t) + B(X_t)X(t-\tau) + C(t), \qquad t \geq 0 \tag{5}$$

where $X = (x_1,\dots,x_n)$ is non-negative, $B(X_t) = [B(t)]$ is a matrix of non-negative functions of $X(w)$ evaluated at past times $w \in [-\tau,t]$, and $C(t) = (I_1,I_2,\dots,I_n)$ is an input function.

Specifically, the system studied was

$$\dot{x}_i(t) = -\alpha x_i(t) + \beta \sum_{k=1}^{n} x_k(t-\tau) y_{ki}(t) + I_i(t),$$

$$y_{jk}(t) = p_{jk} z_{jk}(t) \Big(\sum_{m=1}^{n} p_{jm} z_{jm}(t) \Big)^{-1},$$

$$z_{jk}(t) = [-u z_{jk}(t) + \beta x_j(t-\tau) x_k(t)]\,\theta(p_{jk}), \tag{6}$$

which can be written in vector form

$$\underset{\sim}{U}(t) = f(t, \underset{\sim}{U}(t), \underset{\sim}{U}(t-\tau)), \tag{7}$$

with

$$\underset{\sim}{U} = (x_1, x_2, \dots, x_n, z_{11}, z_{12}, \dots, z_{n,n-1}, z_{nn}),$$

$$F = (f_1, f_2, \dots, f_n, f_{11}, f_{12}, \dots, f_{n,n-1}, z_{nn}),$$

$$f_i = -\alpha x_i + \beta \sum_{k=1}^{n} x_k(t-\tau) p_{ki} z_{ki} \sum_{m=1}^{n} p_{km} z_{km}{}^{-1} + I_i,$$

and

$$f_{jk} = [-uz_{jk} + \beta x_j(t-\tau)x_k]\theta(p_{jk}).$$

Equation (7) corresponds to the system given in Equation (3). By restricting its function somewhat, a machine based on a system like (7) can be taught to predict an event B whenever the event A occurs. This is phrased in another way by saying that we wish to teach the machine the list AB, or, in terms of an idealized human subject H, we wish to teach H the list of letters AB.

It would appear that a system such as (3) is involved in the problem of building a teaching machine. What needs to be specified are the functions f_i which would incorporate such properties as remembering, correcting errors, improving with practice, etc.

(ii) On a Class of Nonlinear Property Filters

(a) Introduction -

In a previous report(20) and article(21) a set of elements q of a set L, a stimulus function $\sigma(q)$, a response function $\rho(q)$, and an interaction function $K_1(p,q)$ are considered to be interrelated in such a manner that the resulting equation integral equation. Both interaction and action in linear systems, as well as discrete linear systems and those with uncountable number of elements were considered and applied to certain specific situations.

It is the purpose of this report to modify the interaction function so that the resulting stimulus-response equation is a nonlinear integral equation. The result should be more in keeping with the fact that in nature we are in general dealing with nonlinear phenomena and such an equation or system thereof would be more representative than linear systems. As is well known, replacing linear systems, about which we know a good deal, with nonlinear systems, about which we know relatively little, presents great mathematical difficulties. We shall consider a class of nonlinear property filters leading to integral equations for which some mathematical theory has been worked out.

(b) Interaction in Nonlinear Systems -

In order to maintain continuity we shall use most of the notation in the report(20). Let $p = (x_1,x_2,\ldots,x_n)$ and $q = (\xi_1,\xi_2,\ldots,\xi_n)$ be any two points of an n-dimensional manifold ε_n and $d\mu_q$ a volume element about the point q. Suppose that an element q of ε_n is subjected to a stimulus $\sigma(q)$ and that the connectivity between each element q of ε_n and all other elements in ε_n has been defined. We denote by $\rho(p)$ the response to a stimulus at p. We shall now assume that the amount of stimulation received by p from q, instead of involving the response ρ linearly, takes it into account in a nonlinear manner. To be specific, we shall assume that the amount of stimulation received by p from q is

$$K_1[p,q;\rho(q)], \tag{1}$$

where K_1 is the so-called interaction function. It should be mentioned that one of the problems here is the delineation of the form of K_1 if it is to reflect the response of an actual physical system. Some experiments here, if possible or feasible, would be of value in determining the analytic character of K_1. Under these assumptions, the total amount of stimulation as p as q encompasses ε_n will be given by the nonlinear integral equation

$$\rho(p) = k\sigma(p) + \lambda\int_{\varepsilon_n} K_1[p,q;\rho(q)]d\mu_q, \quad p,q \in \varepsilon_n, \tag{2}$$

where k is a constant amplifying factor for $\sigma(p)$ and λ is an amplifying factor reflecting the stimulation at p due to the totality of elements q belonging to ε_n. No general theory for the solution of equation (2) with an arbitrary nonlinear kernel K_1 exists. Instead, certain classes of nonlinear integral equations, especially those tied to physical problems, have been examined. For some of these existence and uniqueness theorems have been developed as well as methods of determining solutions. One such class is that discussed in the paper by Hammerstein(22).

(c) A Class of Nonlinear Filters -

Let us consider the class of nonlinear integral equations

$$P(p) + \int_{\varepsilon_n} K)p,q)F[q,P(q)]d\mu_q = 0, \quad p,q \in \varepsilon_n \tag{3}$$

of the Hammerstein type. We assume that the kernel K(p,q) belongs to the class L_2, which implies that the functions

$$\left(\int_{\mathcal{E}_n} K^2(p,q)d\mu_q\right)^{\frac{1}{2}} = \alpha(p), \tag{4}$$

$$\left(\int_{\mathcal{E}_n} K^2(p,q)d\mu_p\right)^{\frac{1}{2}} = \beta(q), \tag{5}$$

exist almost everywhere in ε_n, and their squares are Lebesgue integrable, i.e.,

$$\|K^2(p,q)\| = \int_{\mathcal{E}_n} \alpha^2(p)d\mu_p = \int_{\mathcal{E}_n} \beta^2(q)d\mu_q \leq M^2,$$

where M > 0.

Equation (2) for a kernel of the type we are considering would have the form

$$\rho(p) = k\sigma(p)-\int_{\mathcal{E}_n} K(p,q)\mathcal{F}[q,\rho(q)]d\mu_q = 0, \tag{6}$$

where

$$\lambda K_1[p,q;\rho(q)] = -K(p,q)\mathcal{F}[q,\rho(q)]. \tag{7}$$

A substitution

$$P(p) = \rho(p)-k\sigma(p) \tag{8}$$

transforms (6) into

$$P(p)+\int_{\mathcal{E}_n} K(p,q)\mathcal{F}[q,P(p)+k\sigma(p)d\mu_q = 0, \tag{9}$$

which is of the form of Equation (3).

A principal analytical method of solving (3) or (9) is to employ the classical method of successive approximation scheme by setting

$$P_0(q) \equiv 0 \tag{10}$$

and determining successive functions by means of the relations

$$P_{n+1}(p) = -\int_{\mathcal{E}_n} K(p,q)F[q,P_n(q)]d\mu_q, \; p,q \in \mathcal{E}_n, \tag{11}$$

$n = 0,1,2,3,\ldots$. The convergence of successive functions $P_i(p)$ is not assured, of course, unless certain conditions are satisfied. A generalization of the proof in Tricomi(23) for the one-dimensional case to an n-dimensional space shows that the sequence

$$P_0(p), P_1(p), P_2(p), \ldots, P_n(p), \ldots$$

converges almost everywhere to a solution of (11) if:

(a) $\int_{\mathcal{E}_n} K^2(p,q)d\mu_q = \alpha^2(p)$, exists everywhere in $\mathcal{E}_n$,

(b) $F(q,v)$ satisfies the Lipschitz condition of the form

$$|F(q,v_1)-F(q,v_2)| < \gamma(q)|v_1-v_2|,$$

(c) $F(q,0)$ belongs to L_2,

and

(d) $\int_{\mathcal{E}_n} \alpha^2(p)\gamma^2(p)d\mu_p = M^2 < 1.$

These conditions are fairly stringent. Nevertheless, they are met by a large number of functions. Since these are sufficient conditions and not necessary, the method can still be employed, but the sequence of functions must be examined for convergence and satisfaction of the equation. In cases where the evaluations are tedious and involved, numerical procedures may be employed.

As an example, consider the equation

$$\rho(x) = \int_0^1 K(x,\xi)(1+\rho(\xi))^2 d\xi,$$

where

$$K(x,\xi) = x, \xi < x$$

$$= \xi, \xi \geq x.$$

We define the iteration process by the sequence

$$\rho_0(x) \equiv 0$$

$$\rho_{n+1}(x) = \int_0^1 K(x,\xi)[1+\rho_n(\xi)]^2 d\xi, \; n=0,1,2,\ldots\; .$$

For n = 0 we have

$$\rho_1(x) = \int_0^1 K(x,\xi)d\xi = x\int_0^x d\xi + \int_x^1 \xi d\xi = x\xi\Big|_0^x + \frac{\xi^2}{2}\Big|_x^1$$

$$= x^2+\tfrac{1}{2}-\tfrac{1}{2}x^2 = \tfrac{1}{2}(x^2+1).$$

For n = 1,

$$\rho_2(x) = \int_0^1 K(x,\xi)[1+\tfrac{1}{2}(\xi^2+1)]^2 d\xi = (1/2)^2\int_0^1 K(x,\xi)(\xi^2+3)^2 d\xi$$

$$= (1/2)^2\left\{x\int_0^x (\xi^2+3)^2 d\xi + \int_x^1 \xi(\xi^2+3)^2 d\xi\right\}$$

$$= (1/2)^2\left\{(1/30)x^6+(7/2)x^4+(9/2)x^2+(31/6)\right\}.$$

The iterates $\rho_3(x)$, $\rho_4(x)$, etc., are obtained similarly.

The crucial element in making use of this theory, as has been mentioned, is the determination of the nature of the kernel K(p,q).

References

1. Hays, D. G., "Automatic Language-Data Processing," in Computer Applications in the Behavioral Sciences, H. Borko, ed., Prentice-Hall, Englewood Cliffs, N.J. (1962).

2. Hays, D. G. and T. W. Ziehe, "Studies in Machine Translation--10: Russian Sentence Structure Determination," RM-2538, The Rand Corp., Santa Monica, California (April, 1960).

3. McConlogue, K. and R. F Simmons, "Analyzing English Syntax with a Pattern-Learning Parser," Comm. of ACM, 11, 687 (November, 1965).

4. Gaifman, H., "Dependency Systems and Phrase-Structure Systems," Information and Control, 8, 304 (June, 1965).

5. Weston, P., "Data Structures for Computations within Networks of Relations," in BCL Report 67.2, Biological Computer Laboratory, University of Illinois, Urbana, 126 pp. (1967).

6. Newell, A., et al., Information Processing Language - V, Prentice-Hall, Englewood Cliffs, N. J. (1964).

7. McCarthy, et al., LISP 1.5 Programmer's Manual, MIT Press, Cambridge, Massachusetts (1962).

8. Perlis, A. J. and C. Thornton, "Symbol Manipulation by Threaded Lists," CACM, 3, 195 (1960).

9. Weizenbaum, J., "Symmetric List Processor," CACM, 6, 524 (1963).

10. Roberts, L. G., "Graphical Communication and Control Languages," in Second Congress on Information System Sciences, Hot Springs, Virginia (1964).

11. Knowlton, K. C., "A Programmer's Description of L6," CACM, 9, 616 (1966).

12. Weston, P., "Data Structures for Computations within Networks of Relations," in BCL Report 67.2,Biological Computer Laboratory, University of Illinois, Urbana, 126 pp. (1967).

13. Bouknight, J., Preliminary User's Manual CSL6 CSL7, Coordinated Science Laboratory, University of Illinois, Urbana (1967).

14. Lee, C. Y. and M. C. Paull, "A Content Addressable Distributed Logic Memory with Applications to Information Retrieval," Proceedings of the I.E.E.E., 51, 925 (June,1963).

15. Lee, C. Y., "Intercommunicating Cells--Basis for a Distributed Logic Computer," Proceedings Fall Joint Computer Conference, 22, 130 (December, 1962).

16. Gaines, R. S. and C. Y. Lee, "An Improved Cell Memory," I.E.E.E. Trans. on Electronic Computers, C-17, 10 (January, 1968).

17. Sturman, J. N., "An Iteratively Structured General Purpose Digital Computer," I.E.E.E. Trans. on Electronic Computers, EC-14, 2 (January, 1968).

18. Salton, G., "Progress in Automatic Information Retrieval," I.E.E.E. Spectrum, 90 (August, 1965)

19. Grossberg, S., "A Prediction Theory of Some Non-linear Functional-Differential Equations. 1. Learning of Lists," Journal of Mathematical Analysis and Applications, 21, 643 (March, 1968).

20. Inselberg, A. and H. Von Foerster, Linear Property Filters, Technical Report No. 2, Electrical Engineering Research Laboratory, University of Illinois, Urbana (1962).

21. Von Foerster, H., "Computation in Neural Nets," Currents in Modern Biology, 1 (March, 1967).

22. Hammerstein, A., "Nichtlineare Integralgleichungen nebst Anwendung," Acta Math., 54, 117 (1930).

23. Tricomi, F. G., "Integral Equations," Interscience Publishers, Inc., New York (1957).

(vii) Papers and Reports Published During this Report Period

Kisylia, A. P., *An Association Processor for Information Retrieval*, Report R-390, Coordinated Science Laboratory, University of Illinois, Urbana (August, 1968).

Von Foerster, H., "What is Memory that it may have Hindsight and Foresight as well," presented at *The Future of the Brain Sciences*, *Third International Conference*, Academy of Medicine, New York (May, 1968). (Proceedings in press)

V. ACCOMPLISHMENTS FROM 12/1/68 - 2/28/69

PREFACE

The following pages give a brief account of the activity associated with the study on cognitive memory during the fifth report period from 1 September to 30 November of 1968. Not reported in this account is the preparation of various scientific papers that have been completed during this report period. They will be submitted under separate cover as Special Technical Reports or as publication reprints.

The contribution of members of the Department of Linguistics, of Mathematics, of Anthropology and of Psychology, whose association with this project is by interest and enthusiasm, rather than by contract, is herewith acknowledged with great gratitude.

Particularly appreciated is the participation in the discussion and seminars by Dr. Humberto Maturana, Professor of Biology, University of Chile, Santiago, Chile, who is at present Visiting Professor at the University of Illinois. His contributions to the group in the neurophysiology of cognition will be of lasting significance.

Again I wish to express my thanks to Dr. John Lilly, Miss Margaret Naeser and Miss Alice Miller of the Communications Research Institute in Miami, Florida, who continued the research on the semantic significance of alternates with our group during an extended period of fourteen days from November 25, 1968 to December 9, 1968.

H. Von Foerster

ACCOMPLISHMENTS FROM 12/1/68 - 2/28/69

Table of Contents

Page

Preface..1

Major Activities and Accomplishments During Report Period.......3

1. Research on the "R2" System..................................3

    (i) Question Analysis Techniques............................3
    (ii) Concept Processing.....................................6
    (iii) Context Modeling......................................8
    (iv) Syntactic Processing...................................9

2. Semantic Compiler...15

3. Basic Concepts in Cognition.................................18

    (i) Review...18
    (ii) Present Work..18

4. Associative Processor.......................................20

    (i) Introduction...20
    (ii) Previous Processors...................................21
    (iii) Present Results......................................23

MAJOR ACTIVITIES AND ACCOMPLISHMENTS DURING REPORT PERIOD

1. Research on the "R2" System - K. Biss, R. Burkholder, R.T. Chien, C. Hartmann, D. Lombardi, P. Raynolds, J. Schultz, F. Stahl, T. Woo, D. Yeaney

Further progress has been made on the pilot information system "Rules of the Road," now in its second generation and abbreviated by "R2." All phases of semantic and syntactic processing have been further developed, with special emphasis on semantic modeling and on the retrieval of information from these models.

(i) Question Analysis Techniques

(Chien, Hartmann, Raynolds)

As given in the last progress report,an automatic classification of questions has been developed. Associated with each class we hope to find the most efficient question-answering and information retrieval technique.

Since making our initial breakdown of questions into classes, we have been looking at papers of question answering techniques that would fit in with techniques we have already developed. One technique we have found is Belnaps(1) technique. His argument [as proposed by Harrah(2)] is that every question not only propounds a state of doubt, but also a state of information. For instance:

> What is the maximum speed for an automobile on a highway of Illinois?
>
> State of doubt: I don't know the maximum speed.
>
> State of information: There exist laws regulating the maximum speed for automobiles in the State of Illinois

From the state of information, we get a list of alternatives from which an answer can be derived. In this case the list of alternatives would be:

> One m.p.h. is the maximum speed for an automobile on a highway of Illinois
>
> Two m.p.h. ...
>
> .
>
> .
>
> .
>
> X m.p.h. ...
>
> .
>
> .

The state of doubt makes a request about the list of alternatives. For instance:

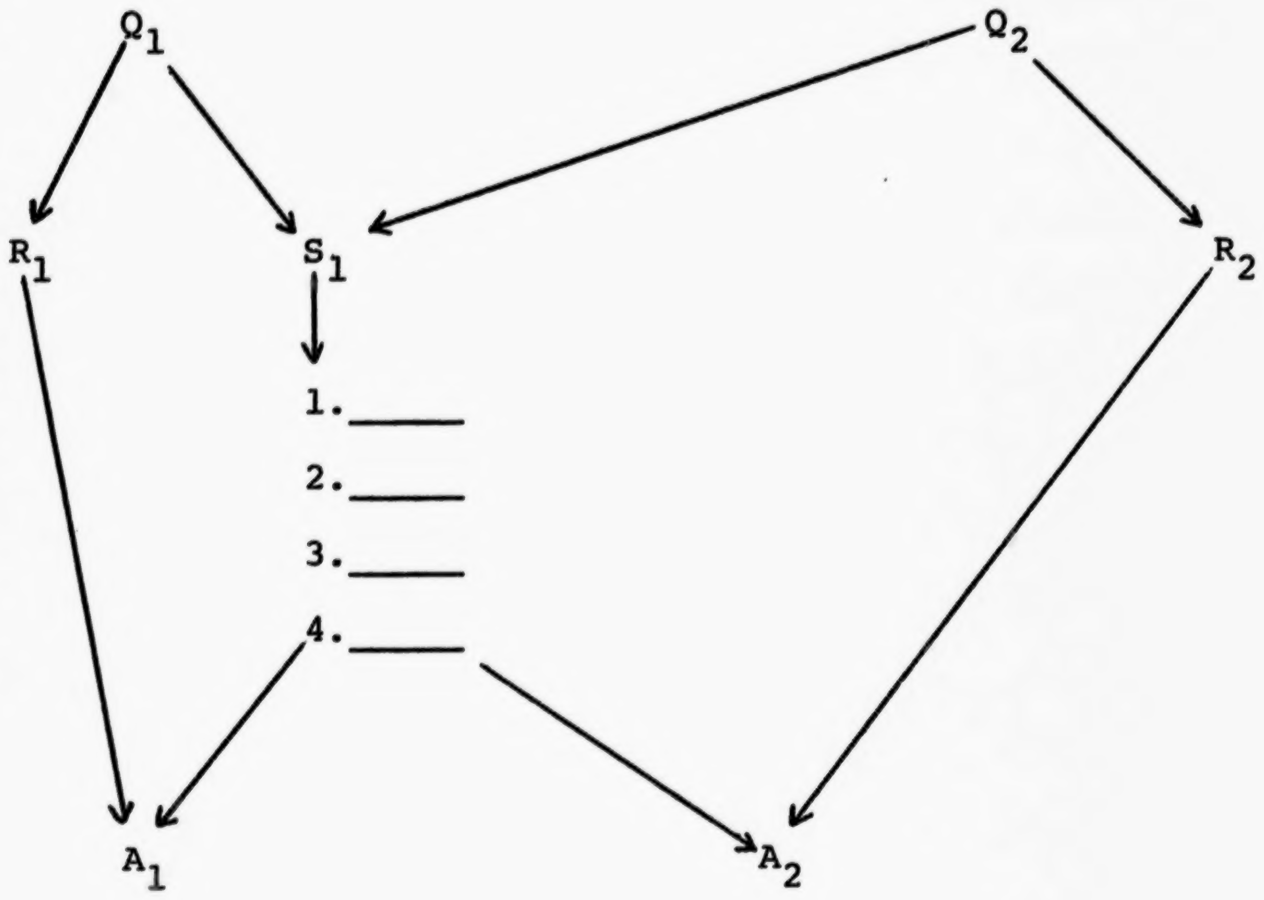
(1) give the unique maximum speed

(2) give any maximum speed

(3) list the maximum speeds

So that every question is uniquely determined by the request it makes and the list of alternatives it presents.

Thus we have



where any question Q_i gives a statement of information S_i with which there is associated a list of alternatives $\{S_i\}$. The request R_i selects some of these alternatives and gives an answer A_i. Thus any questions Q_i and Q_j are equivalent if and only if $R_i = R_j$ and $\{S_i\} = \{S_j\}$. In the above example $R_1 \neq R_2$ so therefore $Q_1 \neq Q_2$. Thus we have a way of telling if two questions are paraphrases of each other. (Q_1 equivalent to $Q_2 \leftrightarrow Q_1$ is a paraphrase of Q_2.)

The question now arises that if $Q_1 \neq Q_2$ could $A_1 = A_2$? This is obviously true for take $R_1 = R_2$ and $\{S_i\}$ = {1.___,2.___, 3.___} $\{S_2\}$ = {2.___,3.___} with $A_1 = A_2$ = 2.___.

As given in previous reports, we break down non yes or no questions into a relative pronoun phrase with the rest of the question called S. S then implies a set of answers which wher intersected with a set given by the relative pronoun part, gives an answer. In general our S contains less information than Belnaps, as our relative pronoun phrase contains more information than his R. In this way we feel that we can come up with a more meaningful intersection.

References

(1) Belnap, Nuel D. Jr., "An Analysis of Questions - Preliminary Report," TM-1287, Systems Dev. Corp. (1963).

(2) Harrah, David, "A Logic of Questions and Answers," Philosophy of Science, 28, 40-46.

(ii) Concept Processing

(Stahl, Woo, Yeaney)

The initial stage in the development of the R2 question-answering information retrieval system with respect to the maximal phrase strategy has been completed. It indicates that information retrieval systems with natural language communication on all levels can be successfully implemented. The initial R2 system has already introduced a number of new features that will aid in the implementation of the total system.

First, the system is data-base independent. That is, the implementation described would have worked equally well on any other coherent textual data-base without any modification to the programs. Strategy 2 with a threshold of 2 has $R_1 2$ 25%.

Second, the maximal phrase concept has defined a new approach to natural language interpretation. The questions posed to the system were not analyzed on a syntactic level. This means there was no attempt made to determine the structure of the question other than the relationship of the constituent maximal phrases to the statements of the data-base. This technique has the obvious advantage of not acquiring the ambiguity involved in syntactic analysis and yet extracting important relationships between the question and the data-base.

There are a number of ways in which the present system may be clearly improved: 1)allowing paragraphs and chapters to be treated as unifying entities, 2) recognition of hyphenated words, 3) replacement capability to accommodate synonyms with possible relational structures utilizing feedback, and 4) sensitivity to pluralization and other variant word forms.

The next phase of development should deal with methods of bringing into play syntactic analysis in the various levels of the system; for interpretation of the question, for structuring the data-base, and for the eventual process of inducing and synthesizing the resultant responses of the system. The addition of this capacity should greatly enhance the ability of the system to deal with natural languages.

The specific points of investigation that seem most promising are the use of locally derived syntactic rules, and the use of "pattern → action" rules as described in Woods(1). Locally derived syntax means mechanically deriving syntactic rules from a given data-base for use in analyzing statements about the content of that data-base.

References

(1) Woods, W.A., "Procedural Semantics for a Question-Answering Machine," AFIPS Conf. Proceedings, 33, Part I, 457-473 (1968).

(iii) Context Modeling

(Chien, Lombardi)

A model has been constructed that is capable of logically structuring the "Rules of the Road." The philosophy behind this model is to create a structure that contains the fundamental topics and relationships in such a manner as to make them accessible to an "uninformed" individual.

Basically the model consists of a number of independent data cells and a list of modifiers that is available to all the data cells.

A data cell is made up primarily of five types of nodes that have the following hierarchical arrangement.

1. MAJOR INDEPENDENT STATIC NODES
2. MAJOR DYNAMIC NODES
3. MAJOR DEPENDENT STATIC NODES
4. MINOR STATIC NODES
5. MINOR DYNAMIC NODES

Briefly, the major nodes contain one word and the minor nodes a sentence or phrase.

Also contained in the data cell are the Relational Data Channels (RDC) that relate the various nodes. These RDC's always go from major to major node or major to minor node. There are four classes of RDC:

1. Operates on a static node and maps into a static node.
2. Operates on a static node and maps into a dynamic node.
3. Operates on a dynamic node and maps into a static node.
4. Operates on a dynamic node and maps into a dynamic node.

Each class of RDC is made up of a number of explicit constituents. For example, the first class has the operator, "specifies a type of an object." Whenever this particular RDC is needed it is simply referred to as ISS, which is the first RDC in the static to static class. Another example is "specifies how an action is accomplished." This is the fourth RDC in the dynamic to dynamic class and is abbreviated 4DD.

Finally the list of modifiers is broken down into three groups: 1) adjectives and adverbs, 2) locations and 3) conditions. Each member of a group has its own address and, consequently, an RDC can refer to a specific member (or members), in relation to a major node, simply by giving that address.

Some analysis has already been done concerning question answering properties of this model and the model seems well suited for this purpose.

(iv) Syntactic Processing

(Biss, Burkholder, Chien, Schultz)

In the last progress report we discussed a dependency grammar parsing program which could parse simple sentences. This program has now been extended to parse compound and complex sentences as well as simple sentences. The program works in the following way. At any word we assume that the word in question is a dependent and we search for its governor, first checking words at a distance one from the dependent, then at a distance two, etc., until a governor is found. Once the first governor for the word in question is found, we make the connection and go to the next word making the assumption that it is now a dependent and search for its governor. If no governor is found for any word, then the word is assumed to be at the head of the dependency tree which underlies the sentence. In this way we produce the most probable parsing of the sentence.

For a more detailed discussion and an evaluation of the program we refer the reader to a Coordinated Science Laboratory report(1) which will soon be printed.

Now that our program is running we will link it in some way to the "semantic analyzer" component to produce the right interpretation of a sentence in a form which the computer can understand.

We have found in developing the grammar for our program that in order to do a good job in parsing sentences we need a relatively complex grammar. For this reason we are looking into the use of a transformational grammar which, theoretically, should be just as powerful as other grammars but much simpler.

A transformational grammar consists of two parts. The first part is an ordinary grammar such as a phrase structure grammar. The second part consists of transforms which map the structure developed in the first part into another structure.

We define a transformation the following way(2): if two or more constructions (or sequences of constructions) which contain the same n classes (whatever else they may contain) occur with the same n-tuples of members of these classes in the same sentence environment, we say that the constructions are transforms of each other and they may be derived from any other of them by a particular transformation. For example:

$$NP_1 \text{ v V } NP_2 \rightarrow NP_2 \text{ v be Ven by } NP_1$$

The police were catching thieves. → Thieves were being caught by the police.

$$NP_1 \text{ v V } NP_2 \text{ X} \leftrightarrow NP_2 \text{ v be Ven X by } NP_1$$

or

$$NP_1 \text{ v be Ven by } NP_1 \text{ X}$$

(X represents whatever follows NP_2 in the transformed secondary sentence.)

The zookeeper spotted the monkey in its cage.→The monkey was spotted in its cage by the zookeeper. ↔The monkey was spotted by the zookeeper in its cage.

Transformations can be 1-1, in the sense that for each individual sentence there is only one transform and conversely (except in the case of homonymity; however, this is eliminated by our immediate constituent analysis and dependency grammar programs); many-one, in that various sentences (with different subjects) have the same transform.

The kernel is the set of elementary sentences and combiners, such that all sentences of the English language are obtained from one or more kernel sentences by means of one or more transformations. Each kernel sentence is a construction of classes. The kernel constructions of English are the following:

NP v V (for V without objects)
NP v V P NP
N v V NP
NP is NP
NP is Adj
NP is P NP
NP is Adv

It is known that each sentence in the English language can be expressed in terms of transformations. We hope to show (in the near future) how we can bring a sentence back to a kernel by means of transformations. For example, we will look at the following:

S_1: Melvin has a happy face.
S_2: Melvin's face is happy.
S_3: Melvin's face is a happy one.
S_4: Melvin's is a happy face.

These are transformed from the kernel sentences:

K_1: Melvin has a face.
K_2: Face is a face.
K_3: Face is happy.

The following transformations are involved:

for S_1: K_3 overlap with K_1

for S_2: K_1, NP has NP→NP's NP, overlap with K_3

for S_3: K_1, NP has NP→NP's NP, overlap with K_2 (first NP)

K_2, PRO of second NP

K_3, overlap with K_2 (second NP)

for S_4: K_1, NP has NP→NP's NP, overlap with K_2 (first NP)

K_3 overlap with K_2(second NP)

Example 2:

S_1: Melvin and Gail went to the party and drank a bottle of pop.

K_1: Melvin went to the party.

K_2: Melvin drank a bottle of pop.

K_3: Gail went to the party.

K_4: Gail drank a bottle of pop.

K_5: A bottle (is) of pop.

At this time we are planning to parse sentences in the following way. We will give an immediate constituent structural description of the sentence using the immediate constituent analysis program which we talked about in the last progress report. We will then scan the sentence to give structural indices of kernels, and scan a kernel list to find if the construction is a kernel. We will then label the kernel, and give references to the kernels that are coordinated and subordinated to it.

R. F. Simmons of the System Development Corporation is working on a kernel approach similar to this but has not been very successful yet.

At this time we are flow charting and writing programs (elementary) that use transformations to seek the kernels and transformations used in the input sentence.

References

(1) Syntactic Analysis in the R2 System (Forthcoming CSL Report).

(2) Harris, Z. E., "Co-occurrences and Transformations in Linguistic Structure," Structure of Language, p. 159.

2. Semantic Compiler - S. M. Taylor, P. Weston

There are in existence today many compilers of various types for the current Babel of programming languages. They are united by the fact that all work strictly with the syntax level of the language, finding in the input text an allowable string of some type and then transforming it into code as a unit independent of its surroundings. This approach works well if the program contains at most syntactic errors which the compiler can recognize as such. If, however, the program contains logical error, if it is a false algorithm, so to speak, then the results are less satisfactory.

This syntactic approach to language has been picked up by some linguists, most notably N. Chomsky and his students, and applied to natural language, with some very interesting results, particularly with respect to an expanded ability to describe the structure of sentences. Most interesting, however, has been the failure of this approach in dealing with problems resulting from the total context imbeddedness of natural language and the meaningful interpretation of utterances under those circumstances. This has led us to the different type of natural language analysis described elsewhere in this section.

In light of the above, it is felt that another cross-fertilization would not be amiss. Therefore, work has started on the application of ideas gained from the study of natural language, and others, to the processing of computer programs.

It is felt that the result could be a higher level debugging aid which would detect a class of logic and design errors outside the capabilities of present compilers, thus forming the basis for an interactive man-machine program composition system.

From the work already done on this, the following conclusions have been made:

1) that a programming language represents an implicit definition of a pseudo-machine (that is: instead of an IBM 360 or a CDC 1604, a FORTRAN computer or an L6 computer) of which machine language is only the most specific case;

2) that a program is a sequence of relationships between the states of this machine;

3) that one can represent these relationships (via cylinder-type data structures) in a way which will allow them to be algorithmically manipulated;

4) that in an interactive system of this type it should not be necessary for the user to have extensive knowledge beyond a familiarity with the programming language he is working with, although some type of (presumably pseudo-English) meta-language for statements about program segments will be necessary. The following action is planned:

1) a program for generating relational structures from a low-level source language (tentatively, a subset of ILLAR, a locally available, sophisticated assembler for CDC 1604 machine language) will be written;

2) a type of flow-chart amenable to computer generation has been designed, and programming will be added to the system to provide a CRT-display of such a flow-chart of the program under analysis;

3) light-pen and typewriter controlled program editing features are planned; as is

4) an option for the execution of the program being analyzed under detailed monitor control.

3. Basic Concepts in Cognition - J. Chow, F. Preparata, S. Ray, B. Wang

(i) Review

Our goal is to demonstrate cognition of a communication by means of directly perceived data combined with imperceivable data, relying upon a complex stored representation of feasible conditions in the environment (the "world-model") from which the communication originates. A simple example may clarify our objective. Suppose the "communication" is a view of a man standing upon a narrow support high above the ground. The human form, the support and a measure of the height above ground are regarded as perceivables. It is the inference that the human may be in danger of a fatal fall and the gravity of the situation which are inferred imperceivables. Cognition consists of identifying combined perceivable and imperceivable factors from which a rational response may be synthesized.

(ii) Present Work

The present reporting period has been occupied by studies of various means of expressing the "world-model" i.e., the collection of inferences which follow from the value of total state (perceivable plus imperceivable variables). If all variables were restricted to binary values and an exhaustive statement of input/output relations were assumed, then the problem reduces to a sequential machine synthesis. The restriction to binary-valued variables, however, appears to us much too confusing. We are proceeding tentatively with each variable having five values: strongly positive, weakly positive, strongly negative, weakly negative, inapplicable (meaningless or undecidable). A model which is sufficiently detailed to produce interesting inferences is being pursued actively.

4. Associative Processor - J. Lipovski, F. P. Preparata

(i) Introduction

One aspect of information retrieval is an investigation of the over all design (architecture) of a computer (processor) which efficiently answers search-type questions, called queries. A conventional computer (Von Neumann Processor) searches for a given item by comparing it with each item to be searched, one at a time. Queries typically involve many searches, some of which depend upon the results of others. Often they take prohibitively long to answer. Considerable effort is being expended to organize the search better on a Von Neumann processor, giving such languages as LISP and such structures as rings or cylinders, but the serial mode of this processor itself slows down the search; this fundamental problem must be solved by considering other processor architectures. These processors search in parallel; they enable a search of all items to be conducted approximately in the time it takes to search one item in a Von Neumann processor. Such processors are called associative processors.

One reason for using associative processors is that they provide fast, real-time, information retrieval. Some industrial, research, and military retrieval problems should be handled in real-time. Fast retrieval may be a convenience, a more powerful tool, or a necessity.

Associative processors may become cost-competitive with Von Neumann Processors for ordinary retrieval of information. They are presently quite expensive, but two points must be considered. Because they are fast, a greater number of queries can be handled in a given time, and the cost per query is important. Further, their design is ideally suited to large scale integration, and the cost of LSI is decreasing.

Lastly, some problems require very large non-numerical processors. But large processors that perform one instruction per time unit may be allocated inefficiently for all but a few programs; large processors that have many specialized modules break down when any module does--this may make a large processor unreliable. A large associative processor is desirable which can be allocated into smaller processors where each processor is of near-optimum size for the process it carries out, and which has few specialized modules and some automated fault detection, testing and repairing of part of the processor while the remainder of the processor runs normally, to improve its reliability.

(ii) Previous Processors

An associative memory is capable of parallel searches, but is run by a conventional computer much like a disk or drum. Early associative memories used cryogenic devices or specially designed core. C. Y. Lee and M. C. Paull gave an associative memory amenable to integrated circuits. D. A. Savitt, H. H. Love and R. E. Troop followed a different philosophy in designing two different processors, called ASP processors, using associative memories. ASP processors seem most suitable for handling the complex queries of information retrieval.

A homogeneous processor, or iterative processor, is a collection of physically identical elements, called cells, which are connected in some regular fashion. The Von Neumann space (not to be confused with the Von Neumann processor mentioned above) and a generalization, the Holland space, are early homogeneous processors. J. Sturman added the concept of homogeneity, from the Von Neumann space, to Lee's memory to obtain an associative processor. This processor is capable of operation if some cells are faulty, if these cells can be detected. J. Smathers considered Sturman's idealized processor in light of practical limitation. One serious practical limitation remains: the communication delay time, and thus the time required for each instruction execution, grows linearly with the number of cells in the processor.

Holland pointed out that iterative processors are able to execute many subroutines simultaneously. This also implies that a very large processor can be allocated into smaller processors, each of near-optimum size. Holland's processor is, however, quite slow and it can become incapable of operation if faulty cells appear.

(iii) Present Results

The processor is a singular tree whose nodes are processor cells, each of which stores a word of memory and has some internal logic, and whose branches are mainly bi-directional channel links. For convenience, we consider trees of uniform degree, d, whose leaves are all at some level, i. Each cell at level i is able to cut its channel link that connects it to the cell at level i-1. With the exception of those channel links that are cut, each cell receives a word of information (perhaps 40 bits) from one of the d+1 channel links connected to it, or from the cell itself, and asynchronously broadcasts this word to the other channel links and to the cell itself. The collection of cells having unbroken channel links between them is a subtree called an elementary processor, and all these channel links form a channel. This processor is easily broken down into elementary processors, each of which can be of nearly optimum size for the process it must carry out.

A well-ordered priority structure exists in hardware in the tree such that for each elementary processor, any set of cells in it has a first (prior) cell. Of the set of cells that intend to read their word of memory into the channel, the prior reads out its contents and drops out of the set. Each cell asynchronously amplifies and broadcasts these contents so that all cells receive them. A clock pulse is then fed through the tree and each cell acts on these contents. All cells receive exactly the same word, which was broadcast from the prior cell, in the same clock cycle. A clock pulse then clears the channel, since the channel has feedback loops that lock onto signals, and repeats the above procedure. Propagation delay in this scheme increases about logarithmically with the number of cells in the processor.

Each tree branch has two unidirectional lines, or rails, so that each cell at level i can signal the cell at level i-1 on one rail and the cell at level i-1 can send one signal to all sibling cells at level i on the other. Each cell at level i is able to cut both rails, to and from the cell at level i-1. The collection of cells having unbroken rails between them is a subtree called a set. One group of rails carries signals from each cell to the root cell of the subtree, and the other group of rails carries a signal from the root cell to all other cells.

A cell is programmed as a data cell or an instruction cell. A collection of one data cell and several instruction cells make up an instruction set. Whenever a word in the channel matches the word (label) in the data cell, it sends a pulse on the rails to all instruction cells; this activates a read flip-flop which puts these instruction cells into the set of all cells which intend to read their word into the channel. By means of the priority structure, each cell reads out in a fixed sequence. By introducing the "label" into the channel, a sequence of instructions will be generated.

A collection of several data cells makes up a data set. A data set may represent a document, for example. A data cell will name the document, other data cells will give some descriptors, others will give references found in the document. Still other cells will be used to store bits which are acquired in processing a query.

One instruction is the MATCH instruction, which has a function and an argument. Any data cell that matches the argument sends a signal on a rail to the root cell. This cell contains a flip-flop, present in all cells but used only in root cells of data sets, containing the state of the set. The root cell interprets the "function" part of the MATCH instruction as a boolean function on the present state and the signal on the rail to get the next state of the set. A sequence of MATCH instructions may choose documents that satisfies a query.

This architecture provides for some fault avoidance. If a cell has a fault, in other than its channel switching, priority structure, and rails, and this is known, then it can be unused, and it might be physically in a subtree without being a member of the instruction or data set. That is to say, no cell is essential to the running of the processor. Using a routine to periodically search for faulty cells, one can obtain a highly reliable processor. Because it contains identical cells, it is amenable to automated testing and repair.

This processor, even on a large scale, appears to have short propagation delays and good reliability. It can be allocated to make smaller elementary processors to be more efficiently used. It appears to be a good design for very large processors as well as for smaller ones. Attention is being given to improving and simplifying some operations, and to finding what other problems might plague large processors.